Supporting Information

Storage of Information using Small Organic Molecules

Brian J. Cafferty,[1†] Alexei S. Ten,[2†] Michael J. Fink,[1] Scott Morey,[2] Daniel J. Preston,[1] Milan Mrksich,[2,3] and George M. Whitesides[1,4,5,*]

[1]Department of Chemistry and Chemical Biology, Harvard University
12 Oxford Street, Cambridge, MA 02138, USA

[2]Department of Chemistry, Northwestern University,
2145 Sheridan Rd, Evanston, IL 60208, USA

[3]Department of Biomedical Engineering, Northwestern University,
2145 Sheridan Rd, Evanston, IL 60208, USA

[4]Kavli Institute for Bionano Science and Technology, Harvard University,
29 Oxford Street, Cambridge, MA 02138, USA

[5]Wyss Institute for Biologically Inspired Engineering,
60 Oxford Street, Cambridge, MA 02138, USA

[†] Authors who contributed equally to this work

*Author to whom correspondence should be addressed:
gwhitesides@gmwgroup.harvard.edu

Table of Contents

12	Program 5: “Image Reconstitution”, reconstitutes a JPEG image from a bitstream	57

Methods

Preparation of solutions of oligopeptides (molbits): Oligopeptides were synthesized using standard Fmoc chemistry on rink-amide resin and purified by HPLC. Stock solutions of each oligopeptide were made in 0.1% TFA with DI water and stored at -20 °C. To prepare the oligopeptides and oligopeptide mixtures for immobilization, each oligopeptide stock solution was distributed into a source plate. Mixing of oligopeptides to form binary data sets was performed using these oligopeptide stock solutions and a Echo® 555 (Labcyte Inc.) liquid handler, with the final concentration of each oligopeptide, when present, at 20 μM (some sequences had to be diluted further to maintain comparable ionization to the other analytes). A Python program written in-house was used to assign oligopeptides from alphanumeric character inputs (translated to ASCII) and bitstrings.

Generating input tables for automated encoding of text: To generate an input table for alphanumeric text for the Echo® 555 liquid handler, a given text was first divided into sections of 6,144 characters (the maximum number of characters that fit on SAMDI 1,536-spot target plate). These blocks of text are then run through a program (“Text Split”, see Supplementary Information for source code), which further divides the 6,144 characters of each block into four sections of 1,536 characters. Each section of 1,536 characters was then assigned to a 384 well plate, with 4 characters (bytes) per well, and a text file (extension .txt) was generated containing the string of characters for each well plate. This file was then used in the program titled "Molbit Encoding". The program also

requires inputs for the volume for each stock solution of oligopeptide to be transferred (in nL), the total capacity per source well (the location of a given oligopeptide to be transferred), the name of the destination plate, and a list of the ASCII binary combinations for each of the characters used (see Table S1 for list of printable ASCII characters and their representation in binary). Once it receives the required inputs, the program matches each character in the .txt file to the appropriate binary ASCII combination and generates an input table for the Echo instrument, including information on source well, transfer volume, destination well, and destination plate name.

Generating input tables for automated encoding of an arbitrary bitstream: To generate an input table for non-ASCII data for the Echo® 555 liquid handler, a bitstream was first generated. The bits were then sequentially numbered 1 through 32. After this process the "Vlookup" function in excel was used to assign a predefined source well for each number. Each group of 32 bits was next assigned with a well of a 1,536-well destination plate. The bitstream, with each entry's associated bit number, source well, and destination well, was then reduced to include only those entries with a bitstream value of 1. Next we used the "Vlookup" function to assign the transfer volume for each entry, based on the source well. Finally, these entries were transferred into a an Echo input table, with information on source well, transfer volume, destination well and destination plate name.

Automated encoding via liquid transfer: Prior to initializing a run on the Echo® 555 liquid handler robot, a source plate (Labcyte Echo Qualified 384-well plates, Cat #: PP-0200) was prepared with the desired oligopeptides to be transferred. Each well of the source plate contained 65 μL of each of the 32 stock solutions (2 mM in oligopeptide).

The number of wells needed for each oligopeptide can be determined from the input table generated via the encoding program. The source plate and destination plate (Greiner Bio-One 384-well plates Cat#: 784201) were placed in storage towers in the Access Laboratory Workstation attached to the liquid handler. To initiate the run, the input table was imported, which defines the locations of the source and destination plates, and the protocol was executed. Once the oligopeptides were transferred, the destination and source plates were covered with lids (Labcyte MicroClime Environmental Microplate Lid Cat#: LL-0310) to ensure that the contents of the plates did not dry.

Preparation of monolayer arrays[21]: Array plates with 384 and 1536 gold spots on steel plates were soaked in a solution of a mixture of EG_3-capped alkane disulfide and a mixed disulfide of EG_3-capped alkanethiol and a maleimide-terminated EG_3-capped alkanethiol for 24 h, at room temperature, to allow formation of a self-assembled monolayer on the gold surface. The solution of disulfides contained an overall concentration of 1 mM of the two monolayer compounds in a stoichiometric ratio (2 to 3) to yield a monolayer wherein the maleimide groups are present at a density of 20%. Following monolayer formation, the plates were soaked in a solution of hexadecyl phosphonic acid (10 mM) for 5 minutes, and rinsed with ethanol, water, ethanol, dried with nitrogen and stored dry under vacuum. SAMDI plates were used within one week of forming monolayers.

Immobilization of peptides onto plates: Prior to immobilization, the peptide mixture plates generated by the Echo® 555 liquid handler were filled with 4 μL of 100 mM Tris buffer at pH 8.0, with a ThermoFisher Multidrop Combi, to ensure the solutions of mixed oligopeptides are at the correct pH and appropriate concentration for conjugation to the monolayer. Each set of four 384-multiwell plates were then transferred to a 1,536-spot

SAMDI plate functionalized with 20% maleimide and displaying a hexadecyl phosphonic acid background between spots. Samples (0.75 μL) from each well of the 384-multiwell plate that contained solution were transferred onto the 1536-spot SAMDI plate utilizing the TECAN Fluent/Freedom Evo instruments, with a MCA 384 head utilizing 15 μL tips, such that each 384-multiwell plate is transferred to one quadrant of a 1536-spot SAMDI plate. In this way the spots are read left to right and top to bottom, and allow the original encoded text to be read. Once transferred, the peptide solutions react with the maleimide groups on the surface of the plate for 10-30 minutes, in a humidified chamber, to covalently immobilize the mixture of peptides. After immobilization, the plate was washed with ethanol, water, ethanol and dried under a stream of nitrogen.

MALDI-TOF MS analysis: SAMDI plates with immobilized oligopeptides were first treated with 2',4',6'-trihydroxyacetophenone matrix solution (THAP, 12 mg/ml in acetone) and then were loaded into an ABSciex TOF-TOF 5800 instrument. Matrix-assisted laser desorption/ionization time-of-flight mass spectra were collected for each spot in positive mode with the instrument setting of 700 shots/spectrum, 5300 laser intensity, stage velocity of 1500 μm/s, 0.61 digitizer setting, and a laser pulse rate of 400 Hz.

Analysis of spectra with program: Prior to analysis of the SAMDI spectra, an input table was generated containing the peptide mass combinations for each of the 95 printable ASCII characters used for each of the 4 bytes (Supplementary Information Table 1). This input table was then divided so that each contained only the peptide combinations for the corresponding byte. This division was done using the "Molbit Decoding" program (source code is available in the Supplementary Information) along with an input of the

95 ASCII characters in quadruplicate, once per byte, and a list of the peptides for each character and byte.

The SAMDI spectra were exported from the instrument computer and analyzed using the "new profiler" program. (The "new profiler" program is available in Supplementary Martials.) This program required the following inputs to run; location of the mass spectrum files, location for the output of generated files, an input table for the byte (1-4) being analyzed, as well as the background threshold. The background threshold is a user-determined value; it is based on the absolute peak intensity relative to the highest peak in the spectrum and is usually set between 20-30%. The background threshold helps avoid false positives in detecting presence of molbits due to the noise in the spectra.

The program functions in the following way. It first scans the spectrum and identifies the maximum intensity value (arbitrary units) and sets this value to 1. It then converts each of the other intensities to relative intensity units based on this parent value. The software then removes any value below the threshold set by the user and generates a new list containing only those peaks remaining above the threshold. Following the generation of the new list, it sums the values of the intensities by rounding to the nearest integer mass value. It then attempts to generate groups of masses based on the two highest consecutive intensity units, followed by single mass intensity groups that cannot be combined. At this point the program scans the input table to find an entry that provides the highest sum of intensities based on mass groups present. Once it finds the entry, it returns the value for the character for which it has decoded. If it fails to match an entry in the input table it will return a "FAILED" response and move on to the next spectrum. Once the software has finished running through the entire dataset, it produces a file that lists the label of the data

spot, the decoded character (if applicable), as well as the masses that were identified for that character. Recovery of information was determined by the number of correctly identified molbits by spectral analysis, divided by the total number of molbits originally encoded, multiplied by 100.

Image compression, encoding, storage, retrieval, and reconstitution: First, if the original copy of an image was larger than the storage space available on one SAMDI 1,536-spot plate (6,144 bytes), that image was compressed, via the JPEG algorithm, to fit on one well plate. The JPEG algorithm was implemented with Adobe Photoshop CS4, version 11.0, with the JPEG quality and blur settings indicated in Supplementary Information Table 2 using the "Save for Web and Devices" function.

After compression, the JPEG files were encoded as bitstreams using the program titled "Image Encoding" (see Supplementary Information for source code), run in Matlab R2015b. The code reads the bytes stored on the local computer hard drive that comprise the JPEG file, and converts these bits to a bitstream. The length of the data contained in the bitstream, in bits, was also read by the code and prepended (as a 16-bit segment) to the front of the bitstream, which was then encoded onto the well plate using the automatic molecular encoding process described above.

Retrieval of data from the well plate was performed as described above, where the output from reading the SAMDI plate was a bitstream. This bitstream, in the form of a text (.txt) file of "1" and "0" with no other characters, was read by a program titled "Image Extraction", which extracts the length of the image file from the first 16 bits of the bitstream and then retrieves that quantity of bits from the bitstream, starting at the 17th

bit (after the string of bits that records the length of the file). This image data was reconstituted into an image file in JPEG format which can be interpreted and displayed by a computer. The error rate during retrieval and reconstitution of each image is included in Supplementary Table 2.

Table S1. Assignment of 32 peptides to 32 molbits (4 molbytes) by ascending mass-to-charge ratio.

Molbyte Number	Molbit Number	Oligopeptide Number	sequence	m.w. (g mol^{-1})	m/z obs.* (a.m.u.)
I	1	**1**	Ac-AK$^{(me3)}$C	404	1255
	2	**2**	Ac-(abu)K$^{(me3)}$C	418	1269
	3	**3**	Ac-VK$^{(me3)}$C	432	1283
	4	**4**	Ac-GGK$^{(me3)}$C	447	1298
	5	**5**	Ac-GVK$^{(me3)}$C	489	1340
	6	**6**	Ac-GLK$^{(me3)}$C	503	1354
	7	**7**	Ac-ALK$^{(me3)}$C	517	1368
	8	**8**	Ac-GFK$^{(me3)}$C	537	1388
II	1	**9**	Ac-GVGK$^{(me3)}$C	546	1397
	2	**10**	Ac-GLGK$^{(me3)}$C	560	1411
	3	**11**	Ac-GAGGK$^{(me3)}$C	575	1426
	4	**12**	Ac-GL(abu)K$^{(me3)}$C	588	1439
	5	**13**	Ac-GFGK$^{(me3)}$C	594	1445
	6	**14**	Ac-GRGK$^{(me3)}$C	603	1454
	7	**15**	Ac-GPAGK$^{(me3)}$C	615	1466
	8	**16**	Ac-AYGK$^{(me3)}$C	624	1475
III	1	**17**	Ac-GPFK$^{(me3)}$C	634	1485
	2	**18**	Ac-GVVGK$^{(me3)}$C	645	1496
	3	**19**	Ac-G(abu)FGK$^{(me3)}$C	679	1530
	4	**20**	Ac-GVFGK$^{(me3)}$C	693	1544
	5	**21**	Ac-GVYGK$^{(me3)}$C	709	1560
	6	**22**	Ac-GARGGK$^{(me3)}$C	731	1582
	7	**23**	Ac-GAVV(abu)K$^{(me3)}$C	744	1595
	8	**24**	Ac-GFYGK$^{(me3)}$C	757	1608
IV	1	**25**	Ac-GYYGK$^{(me3)}$C	773	1624
	2	**26**	Ac-GYYAK$^{(me3)}$C	787	1638
	3	**27**	Ac-GPYFK$^{(me3)}$C	797	1648
	4	**28**	Ac-GRGFGK$^{(me3)}$C	807	1658
	5	**29**	Ac-GYFGGK$^{(me3)}$C	814	1665
	6	**30**	Ac-GYYGGK$^{(me3)}$C	830	1681
	7	**31**	Ac-AYYGGK$^{(me3)}$C	844	1695
	8	**32**	Ac-GYY(abu)GK$^{(me3)}$C	858	1709

Ac: acetyl, Abu: 2-aminobutyric acid, A: alanine, G: glycine, C: cysteine, F: phenylalanine, K$^{(me3)}$: N$^{\varepsilon}$,N$^{\varepsilon}$,N$^{\varepsilon}$-trimethyl lysine L: leucine, P: proline, R: arginine, V: valine, Y: tyrosine, a.m.u.: atomic mass unit.

*The observed mass-to-charge ratio (m/z) includes the oligopeptide plus the mixed disulfide derived from the SAM, which is formed during laser-assisted desorption/ionization (+ 851 a.m.u.).

Table S2. List of printable characters encoded using mixtures of molbits and their 8-bit ASCII character code*

A	01000001	Q	01010001	g	01100111	w	01110111	!	00100001	;	00111011
B	01000010	R	01010010	h	01101000	x	01111000	“	00100010	<	00111100
C	01000011	S	01010011	i	01101001	y	01111001	#	00100011	=	00111101
D	01000100	T	01010100	j	01101010	z	01111010	$	00100100	>	00111110
E	01000101	U	01010101	k	01101011	0	00110000	%	00100101	?	00111111
F	01000110	V	01010110	l	01101100	1	00110001	&	00100110	[	01011011
G	01000111	W	01010111	m	01101101	2	00110010	‘	00100111	\	01011100
H	01001000	X	01011000	n	01101110	3	00110011	(	00101000	]	01011101
I	01001001	Y	01011001	o	01101111	4	00110100	)	00101001	^	01011110
J	01001010	Z	01011010	p	01110000	5	00110101	*	00101010	_	01011111
K	01001011	a	01100001	q	01110001	6	00110110	+	00101011	`	01100000
L	01001100	b	01100010	r	01110010	7	00110111	,	00101100	{	01111011
M	01001101	c	01100011	s	01110011	8	00111000	-	00101101	|	01111100
N	01001110	d	01100100	t	01110100	9	00111001	.	00101110	}	01111101
O	01001111	e	01100101	u	01110101	“space”	00100000	/	00101111	~	01111110
P	01010000	f	01100110	v	01110110	@	01000000	:	00111010		

* These are the characters that *have* been written, not all possible character that *can* be written.

Table S3. Properties of images and JPEG conversion parameters.

Image name	Compressed Dimensions (px)	JPEG Quality	JPEG Blur	Compressed Size (bits(b) /bytes(B))	Retrieval Error
Claud Shannon	150 x 150	49	0.2	48,752b/6,094B	0.00 %
The Great Wave off Kanagawa	240 x 200	11	0.2	47,624b/5,953B	0.00 %

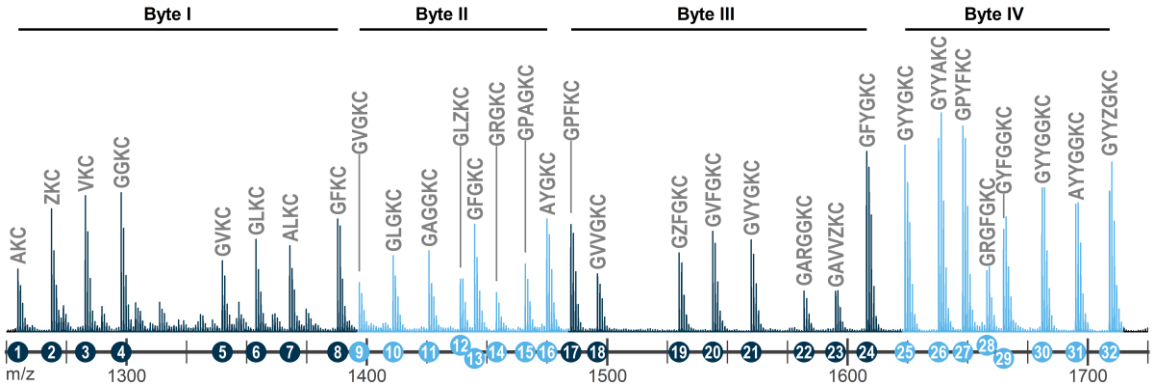

Figure S1. Spectrum of all 32 oligopeptides examined in this study. Oligopeptides were grouped by molecular weight into sets of eight, representing a byte of information (4 bytes total). The single-letter codes of residues in the information region are listed above each peak in the mass spectrum.

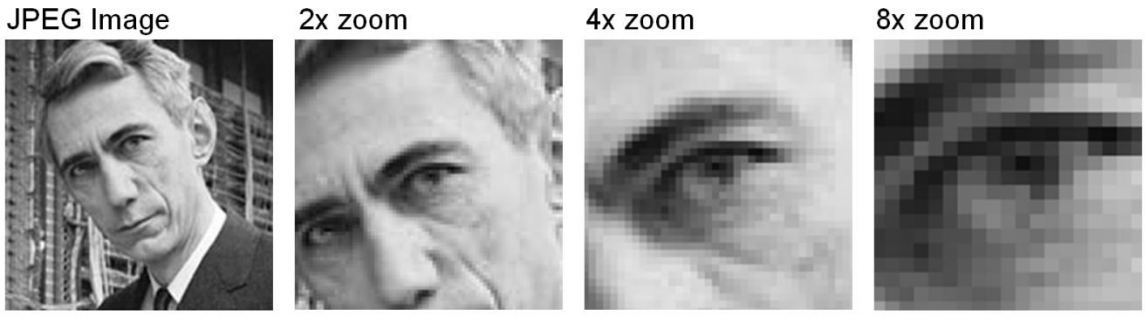

Figure S2. Losses during compression of an image to a JPEG file. The original and reconstituted JPEG images exhibit pixilation that is more visible at high magnification, a consequence of the compression required to reduce the image size.

Text of Feynman's "There is plenty of room at the bottom" written using 32-molbit encoding strategy on 7 MALDI target plates. Characters (bytes) that were misidentified are in red, underlined and bolded. 301 bits of 306,504 total bits (0.01%) were misidentified.

The storage of this text was carried out only once. Most of the errors were in Plate 4, and reflect (we infer) some defect in manipulating this plate.

Plate 1

Plenty of Room at the Bottom Richard P. Feynman (Dated: Dec. 1959). I imagine experimental physicists must often look with envy at men like Kamerlingh Onnes, who discovered a field like low temperature, which seems to be bottomless and in which one can go down and down. Such a man is then a leader and has some temporary monopoly in a scientific adventure. Percy Bridgman, in designing a way to obtain higher pressures, opened up another new field and was able to move into it and to lead us all along. The development of ever higher vacuum was a continuing development of the same kind. I would like to describe a field, in which little has been done, but in which an enormous amount can be done in principle. This field is not quite the same as the others in that it will not tell us much of fundamental physics (in the sense of, "What are the strange particles?") but it is more like solid-state physics in the sense that it might tell us much of great interest about the strange phenomena that occur in complex situations. Furthermore, a point that is most important is that it would have an enormous number of technical applications. What I want to talk about is the problem of manipulating and controlling things on a small scale. As soon as I mention this, people tell me about miniaturization, and how far it has progressed today. They tell me about electric motors that are the size of the nail on your small finger. And there is a device on the market, they tell me, by which you can write the Lord's Prayer on the head of a pin. But that's nothing; that's the most primitive, halting step in the direction I intend to discuss. It is a staggeringly small world that is below. In the year 2000, when they look back at this age, they will wonder why it was not until the year 1960 that anybody began seriously to move in this direction. Why cannot we write the entire 24 volumes of the Encyclopedia Brittanica on the head of a pin? Let's see what would be involved. The head of a pin is a sixteenth of an inch across. If you magnify it by 25,000 diameters, the area of the head of the pin is then equal to the area of all the pages of the Encyclopedia Brittanica. Therefore, all it is necessary to

do is to reduce in size all the writing in the Encyclopedia by 25,000 times. Is that possible? The resolving power of the eye is about 1/120 of an inch-that is roughly the diameter of one of the little dots on the fine half-tone reproductions in the Encyclopedia. This, when you demagnify it by 25,000 times, is still 80 angstroms in diameter-32 atoms across, in an ordinary metal. In other words, one of those dots still would contain in its area 1,000 atoms. So, each dot can easily be adjusted in size as required by the photoengraving, and there is no question that there is enough room on the head of a pin to put all of the Encyclopedia Brittanica. Furthermore, it can be read if it is so written. Let's imagine that it is written in raised letters of metal; that is, where the black is in the Encyclopedia, we have raised letters of metal that are actually 1/25,000 of their ordinary size. How would we read it? If we had something written in such a way, we could read it using techniques in common use today. (They will undoubtedly find a better way when we do actually have it written, but to make my point conservatively I shall just take techniques we know today.) We would press the metal into a plastic material and make a mold of it, then peel the plastic off very carefully, evaporate silica into the plastic to get a very thin film, then shadow it by evaporating gold at an angle against the silica so that all the little letters will appear clearly, dissolve the plastic away from the silica film, and then look through it with an electron microscope! There is no question that if the thing were reduced by 25,000 times in the form of raised letters on the pin, it would be easy for us to read it today. Furthermore; there is no question that we would find it easy to make copies of the master; we would just need to press the same metal plate again into plastic and we would have another copy. How do we write small? The next question is: How do we write it? We have no standard technique to do this now. But let me argue that it is not as difficult as it first appears to be. We can reverse the lenses of the electron microscope in order to demagnify as well as magnify. A source of ions, sent through the microscope lenses in reverse, could be focused to a very small spot. We could write with that spot like we write in a TV cathode ray oscilloscope, by going across in lines, and having an adjustment which determines the amount of material which is going to be deposited as we scan in lines. This method might be very slow because of space charge limitations. There will be more rapid methods. We could first m**ake,** perhaps by some photo process, a screen which has holes in it in the form of the letters. Then we would strike an arc behind

the holes and draw metallic ions through the holes; then we could again use our system of lenses and make a small image in the form of ions, which would deposit the metal on the pin. A simpler way might be this (though I am not sure it 2 would work): We take light and, through an optical microscope running backwards, we focus it onto a very small photoelectric screen. Then electrons come away from the screen where the light is shining. These electrons are focused down in size by the electron microscope lenses to impinge directly upon the surface of the metal. Will such a beam etch away the metal if it is run long enough? I don't know. If it doesn't work for a metal surface, it must be possible to find some surface with which to coat the original pin so that, where the electrons bombard, a change is made which we could recognize later. There is no intensity problem in these devices-not what you are used to in magnification, where you have to take a few electrons and spread them over a bigger and bigger screen; it is just the opposite. The light which we get from a page is concentrated onto a very small area so it is very intense. The few electrons which com

Plate 2

e from the photoelectric screen are demagnified down to a very tiny area so that, again, they are very intense. I don't know why this hasn't been done yet! That's the Encyclopedia Brittanica on the head of a pin, but let's consider all the books in the world. The Library of Congress has approximately 9 million volumes; the British Museum Library has 5 million volumes; there are also 5 million volumes in the National Library in France. Undoubtedly there are duplications, so let us say that there are some 24 million volumes of interest in the world. What would happen if I print all this down at the scale we have been discussing? How much space would it take? It would take, of course, the area of about a million pinheads because, instead of there being just the 24 volumes of the Encyclopaedia, there are 24 million volumes. The million pinheads can be put in a square of a thousand pins on a side, or an area of about 3 square yards. That is to say, the silica replica with the paper-thin backing of plastic, with which we have made the copies, with all this information, is on an area of approximately the size of 35 pages of the Encyclopedia. That is about half as many pages as there are in this magazine. All of the information which all of mankind has every recorded in books can be carried around in a

pamphlet in your hand-and not written in code, but a simple reproduction of the original pictures, engravings, and everything else on a small scale without loss of resolution. What would our librarian at Caltech say, as she runs all over from one building to another, if I tell her that, ten years from now, all of the information that she is struggling to keep track of- 120,000 volumes, stacked from the floor to the ceiling, drawers full of cards, storage rooms full of the older books-can be kept on just one library card! When the University of Brazil, for example, finds that their library is burned, we can send them a copy of every book in our library by striking off a copy from the master plate in a few hours and mailing it in an envelope no bigger or heavier than any other ordinary air mail letter. Now, the name of this talk is "There is Plenty of Room at the Bottom"-not just "There is Room at the Bottom." What I have demonstrated is that there is room- that you can decrease the size of things in a practical way. I now want to show that there is plenty of room. I will not now discuss how we are going to do it, but only what is possible in principle-in other words, what is possible according to the laws of physics. I am not inventing anti-gravity, which is possible someday only if the laws are not what we think. I am telling you what could be done if the laws are what we think; we are not doing it simply because we haven't yet gotten around to it. - Information on a small scale - Suppose that, instead of trying to reproduce the pictures and all the information directly in its present form, we write only the information content in a code of dots and dashes, or something like that, to represent the various letters. Each letter represents six or seven "bits" of information; that is, you need only about six or seven dots or dashes for each letter. Now, instead of writing everything, as I did before, on the surface of the head of a pin, I am going to use the interior of the material as well. Let us represent a dot by a small spot of one metal, the next dash, by an adjacent spot of another metal, and so on. Suppose, to be conservative, that a bit of information is going to require a little cube of atoms 5 times 5 times 5-that is 125 atoms. Perhaps we need a hundred and some odd atoms to make sure that the information is not lost through diffusion, or through some other process. I have estimated how many letters there are in the Encyclopaedia, and I have assumed that each of my 24 million books is as big as an Encyclopaedia volume, and have calculated, then, how many bits of information there are (1015). For each bit I allow 100 atoms. And it turns out that all of the information that man has carefully

accumulated in all the books in the world can be written in this form in a cube of material one two-hundredth of an inch wide- which is the barest piece of dust that can be made out by the human eye. So there is plenty of room at the bottom! Don't tell me about microfilm! This fact-that enormous amounts of information can be carried in an exceedingly small space-is, of course, well known to the biologists, and resolves the mystery which existed before we understood all this clearly, of how it could be that, in the tiniest cell, all of the information for the organization of a complex creature such as ourselves can be stored. All this information-whether we have brown eyes, or whether we think at all, or that in the embryo the jawbone should first develop with a little hole in the side so that later a nerve can grow through it-all this information is contained in a very tiny fraction of the cell in the form of long-chain DNA molecules in which approximately 50 atoms are used for one bit of 3 information about the cell. -Better electron microscopes - If I have written in a code, with 5 times 5 times 5 atoms to a bit, the question is: How could I read it today? The electron microscope is not quite good enough, with the greatest care and effort, it can only resolve about 10 angstroms. I would like to try and impress upon you while I am talking about all of these things on a small scale, the importance of improving the electron microscope by a hundred times. It is not impossible; it is not against the laws of diffraction of the electron. The wave length of the electron in such a microscope is only 1/20 of an angstrom. So it should be possible to see the individual atoms. What good would it be to see individual atoms distinctly? We have friends in other fields-in biology, for instance. We physicists often look at them and say, "You know the reason you fellows are making so little progress?" (Actually I don't know any field where they are making more rapid progress than they are in biology today.) "You should use more mathematics, like we do." They coul

Plate 3

d answer us-but they're polite, so I'll answer for them: "What you should do in order for us to make more rapid progress is to make the electron microscope 100 times better." What are the most central and fundamental problems of biology today? They are questions like: What is the sequence of bases in the DNA? What happens when you have a mutation? How is the base order in the DNA connected to the order of amino acids in

the protein? What is the structure of the RNA; is it single-chain or double-chain, and how is it related in its order of bases to the DNA? What is the organization of the microsomes? How are proteins synthesized? Where does the RNA go? How does it sit? Where do the proteins sit? Where do the amino acids go in? In photosynthesis, where is the chlorophyll; how is it arranged; where are the carotenoids involved in this thing? What is the system of the conversion of light into chemical energy? It is very easy to answer many of these fundamental biological questions; you just look at the thing! You will see the order of bases in the chain; you will see the structure of the microsome. Unfortunately, the present microscope sees at a scale which is just a bit too crude. Make the microscope one hundred times more powerful, and many problems of biology would be made very much easier. I exaggerate, of course, but the biologists would surely be very thankful to you- and they would prefer that to the criticism that they should use more mathematics. The theory of chemical processes today is based on theoretical physics. In this sense, physics supplies the foundation of chemistry. But chemistry also has analysis. If you have a strange substance and you want to know what it is, you go through a long and complicated process of chemical analysis. You can analyze almost anything today, so I am a little late with my idea. But if the physicists wanted to, they could also dig under the chemists in the problem of chemical analysis. It would be very easy to make an analysis of any complicated chemical substance; all one would have to do would be to look at it and see where the atoms are. The only trouble is that the electron microscope is one hundred times too poor. (Later, I would like to ask the question: Can the physicists do something about the third problem of chemistry-namely, synthesis? Is there a physical way to synthesize any chemical substance? The reason the electron microscope is so poor is that the f- value of the lenses is only 1 part to 1,000; you don't have a big enough numerical aperture. And I know that there are theorems which prove that it is impossible, with axially symmetrical stationary field lenses, to produce an f-value any bigger than so and so; and therefore the resolving power at the present time is at its theoretical maximum. But in every theorem there are assumptions. Why must the field be symmetrical? I put this out as a challenge: Is there no way to make the electron microscope more powerful? - The marvelous biological system - The biological example of writing information on a small scale has inspired me to think of something that should

be possible. Biology is not simply writing information; it is doing something about it. A biological system can be exceedingly small. Many of the cells are very tiny, but they are very active; they manufacture various substances; they walk around; they wiggle; and they do all kinds of marvelous things-all on a very small scale. Also, they store information. Consider the possibility that we too can make a thing very small which does what we want-that we can manufacture an object that maneuvers at that level! There may even be an economic point to this business of making things very small. Let me remind you of some of the problems of computing machines. In computers we have to store an enormous amount of information. The kind of writing that I was mentioning before, in which I had everything down as a distribution of metal, is permanent. Much more interesting to a computer is a way of writing, erasing, and writing something else. (This is usually because we don't want to waste the material on which we have just written. Yet if we could write it in a very small space, it wouldn't make any difference; it could just be thrown away after it was read. It doesn't cost very much for the material). - Miniaturizing the computer - I don't know how to do this on a small scale in a practical way, but I do know that computing machines are very large; they fill rooms. Why can't we make them very small, make them of little wires, little elements-and by little, I mean little. For instance, the wires should be 10 or 100 atoms in diameter, and the circuits should be a few thousand angstroms across. Everybody who has analyzed the logical theory of computers has come to the 4 conclusion that the possibilities of computers are very interesting-if they could be made to be more complicated by several orders of magnitude. If they had millions of times as many elements, they could make judgments. They would have time to calculate what is the best way to make the calculation that they are about to make. They could select the method of analysis which, from their experience, is better than the one that we would give to them. And in many other ways, they would have new qualitative features. If I look at your face I immediately recognize that I have seen it before. (Actually, my friends will say I have chosen an unfortunate example here for the subject of this illustration. At least I recognize that it is a man and not an apple.) Yet there is no machine which, with that speed, can take a picture of a face and say even that it is a man; and much less that it is the same man that you showed it before-unless it is exactly the same pict**u**re. If the face is changed; if I am closer to the face; if I am further from the

face; if the light changes-I recognize it anyway. Now, this little computer I carry in my head is easily able to do that. The computers that we build are not able to do that. The number of elements in this bone box of mine are enormously greater than the number of elements in our "wonderful" com

Plate 4

puters. But our mechanical computers are too big; the elements in this box are microscopic. I want to make some that are submicroscopic. If we wanted to make a computer that had all these marvelous extra qualitative abilities, we would have to make it, perhaps, the size of the Pentagon. This has several disadvantages. First, it requires too much material; there may not be enough germanium in the world for all the transistors which would have to be put into this enormous thing. There is also the problem of heat generation and power consumption; TVA would be needed to run the computer. But an even more practical difficulty is that the computer would be limited to a certain speed. Because of its large size, there is finite time required to get the information from one place to another. The information cannot go any faster than the speed of light-so, ultimately, when our computers get faster and faster and more and more elaborate, we will have to make them smaller and smaller. But there is plenty of room to make them smaller. There is nothing that I can see in the physical laws that says the computer elements cannot be made enormously smaller than they are now. In fact, there may be certain advantages. - Miniaturization by evaporation - How can we make such a device? What kind of manufacturing processes would we use? One possibility we might consider, since we have talked about writing by putting atoms down in a certain arrangement, would be to evaporate the material, then evaporate the insulator next to it. Then, for the next layer, evaporate another position of a wire, another insulator, and so on. So, you simply evaporate until you have a block of stuff which has the elements- coils and condensers, transistors and so on-of exceedingly fine dimensions. But I would like to discuss, just for amusement, that there are other possibilities. Why can't we manufacture these small computers somewhat like we manufacture the big ones? Why can't we drill holes, cut things, solder things, stamp things out, mold different shapes all at an infinitesimal level? What are the limitations as to how small a thing has to be before you

can no longer mold it? How many times when you are working on something frustratingly tiny like your wife's wristwatch, have you said to yourself, "If I could only train an ant to do this!" What I would like to suggest is the possibility of training an ant to train a mite to do this. What are the possibilities of small but movable machines? They may or may not be useful, but they surely would be fun to make. Consider any machine- for example, an automobile- and ask about the problems of making an infinitesimal machine like it. Suppose, in the particular design of the automobile, we need a certain precision of the parts; we need an accuracy, let's suppose, of 4/10,000 of an inch. If things are more inaccurate than that in the shape of the cylinder and so on, it isn't going to work very well. If I make the thing too small, I have to worry about the size of the atoms; I can't make a circle of "balls" so to speak, if the circle is too small. So, if I make the error, corresponding to 4/10,000 of an inch, correspond to an error of 10 atoms, it turns out that I can reduce the dimensions of an automobile 4,000 times, approximately-so that it is 1 mm. across. Obviously, if you redesign the car so that it would work with a much larger tolerance, which is not at all impossible, then you could make a much smaller device. It is interesting to consider what the problems are in such small machines. Firstly, with parts stressed to the same degree, the forces go as the area you are reducing, so that things like weight and inertia are of relatively no importance. The strength of material, in other words, is very much greater in proportion. The stresses and expansion of the flywheel from centrifugal force, for example, would be the same proportion only if the rotational speed is increased in the same proportion as we decrease the size. On the other hand, the metals that we use have a grain structure, and this would be very annoying at small scale because the material is not homogeneous. Plastics and glass and things of this amorphous nature are very much more homogeneous, and so we would have to make our machines out of such materials. There are problems associated with the electrical part of the system-with the copper wires and the magnetic parts. The magnetic properties on a very small scale are not the same as on a large scale; there is the "domain" problem involved. A big magnet made of millions 5 of domains can only be made on a small scale with one domain. The electrical equipment won't simply be scaled down; it has to be redesigned. But I can see no reason why it can't be redesigned to work again. - Problems of lubrication - Lubrication involves some interesting points. The effective viscosity of

oil would be higher and higher in proportion as we went down (and if we increase the speed as much as we can). If we don't increase the speed so much, and change from oil to kerosene or some other fluid, the problem is not so bad. But actually we may not have to lubricate at all! We have a lot of extra force. Let the bearings run dry; they won't run hot because the heat escapes away from such a small device very, very rapidly. This rapid heat loss would prevent the gasoline from exploding, so an internal combustion engine is impossible. Other chemical reactions, liberating energy when cold, can be used. Probably an external supply of electrical power would be most convenient for such small machines. What would be the utility of such machines? Who knows? Of course, a small automobile would only be useful for the mites to drive around in, and I suppose our Christian interests don't go that far. However, we did note the possibility of the manufacture of small elements for computers in completely automatic factories, containing lathes and other machine tools at the very small level. The small lathe would not have to be exactly like our big lathe. I leave to your imagination the improvement of the design to take full advantage of the properties o

Plate 5

f things on a small scale, and in such a way that the fully automatic aspect would be easiest to manage. A friend of mine (Albert R. Hibbs) suggests a very interesting possibility for relatively small machines. He says that, although it is a very wild idea, it would be interesting in surgery if you could swallow the surgeon. You put the mechanical surgeon inside the blood vessel and it goes into the heart and "looks" around. (Of course the information has to be fed out.) It finds out which valve is the faulty one and takes a little knife and slices it out. Other small machines might be permanently incorporated in the body to assist some inadequately functioning organ. Now comes the interesting question: How do we make such a tiny mechanism? I leave that to you. However, let me suggest one weird possibility. You know, in the atomic energy plants they have materials and machines that they can't handle directly because they have become radioactive. To unscrew nuts and put on bolts and so on, they have a set of master and slave hands, so that by operating a set of levers here, you control the "hands" there, and can turn them this way and that so you can handle things quite nicely. Most of

these devices are actually made rather simply, in that there is a particular cable, like a marionette string, that goes directly from the controls to the "hands." But, of course, things also have been made using servomotors, so that the connection between the one thing and the other is electrical rather than mechanical. When you turn the levers, they turn a servomotor, and it changes the electrical currents in the wires, which repositions a motor at the other end. Now, I want to build much the same device-a master slave system which operates electrically. But I want the slaves to be made especially carefully by modern large-scale machinists so that they are one-fourth the scale of the "hands" that you ordinarily maneuver. So you have a scheme by which you can do things at one-quarter scale anyway-the little servo motors with little hands play with little nuts and bolts; they drill little holes; they are four times smaller. Aha! So I manufacture a quarter-size lathe; I manufacture quarter-size tools; and I make, at the one-quarter scale, still another set of hands again relatively one-quarter size! This is one-sixteenth size, from my point of view. And after I finish doing this I wire directly from my large-scale system, through transformers perhaps, to the one-sixteenth-size servomotors. Thus I can now manipulate the one-sixteenth size hands. Well, you get the principle from there on. It is rather a difficult program, but it is a possibility. You might say that one can go much farther in one step than from one to four. Of course, this has all to be designed very carefully and it is not necessary simply to make it like hands. If you thought of it very carefully, you could probably arrive at a much better system for doing such things. If you work through a pantograph, even today, you can get much more than a factor of four in even one step. But you can't work directly through a pantograph which makes a smaller pantograph which then makes a smaller pantograph-because of the looseness of the holes and the irregularities of construction. The end of the pantograph wiggles with a relatively greater irregularity than the irregularity with which you move your hands. In going down this scale, I would find the end of the pantograph on the end of the pantograph on the end of the pantograph shaking so badly that it wasn't doing anything sensible at all. At each stage, it is necessary to improve the precision of the apparatus. If, for instance, having made a small lathe with a pantograph, we find its lead screw irregular-more irregular than the large-scale one-we could lap the lead screw against breakable nuts that you can reverse in the usual way back and forth until this lead screw is, at its scale, as accurate as

our original lead screws, at our scale. We can make flats by rubbing unflat surfaces in triplicates together-in three pairs-and the flats then become flatter than the thing you started with. Thus, it is not impossible to improve precision on a small scale by the correct operations. So, when we build this stuff, it is necessary at each step to improve the accuracy of the equipment by working for awhile down there, making accurate lead screws, Johansen blocks, and all the other materials which we use in accurate machine work at the higher level. We have to stop at each level and manufacture all the stuff to go to the next level-a very long and very difficult program. Perhaps you can figure a better way than that to get down to small scale more rapidly. Yet, after all this, you have just got one little baby lathe four thousand times smaller than usual. But we were thinking of making an enormous computer, which we were going to build by drilling holes on this lathe to make little washers for the computer. How many washers can you manufacture on this one lathe? - A hundred tiny hands - When I make my first set of slave "hands" at one fourth scale, I am going to make ten sets. I make ten sets of "hands," and I wire them to my original levers so they each do exactly the same thing at the same time in parallel. Now, when I am making my new devices one quarter again as small, I let each one manufacture ten copies, so that I would have a hundred "hands" at the 1/16th size. Where am I going to put the million lathes that I am going to have? Why, there is nothing to it; the volume is much less than that of even one full-scale lathe. For instance, if I made a billion little lathes, each 1/4000 of the scale of a regular lathe, there are plenty of materials and space available because in the billion little ones there is less than 2 percent of the materials in one big lathe. It doesn't cost anything for materials, you see. So I want to build a billion tiny factories, models of each other, which are manufacturing simultaneously, drilling holes, stamping parts, and so on. As we go down in size, there are a number of interesting problems that arise. All things do not

Plate 6

simply scale down in proportion. There is the problem that materials stick together by the molecular (Van der Waals) attractions. It would be like this: After you have made a part and you unscrew the nut from a bolt, it isn't going to fall down because the gravity isn't appreciable; it would even be hard to get it off the bolt. It would be like those old movies

of a man with his hands full of molasses, trying to get rid of a glass of water. There will be several problems of this nature that we will have to be ready to design for. - Rearranging the atoms - But I am not afraid to consider the final question as to whether, ultimately-in the great future-we can arrange the atoms the way we want; the very atoms, all the way down! What would happen if we could arrange the atoms one by one the way we want them (within reason, of course; you can't put them so that they are chemically unstable, for example). Up to now, we have been content to dig in the ground to find minerals. We heat them and we do things on a large scale with them, and we hope to get a pure substance with just so much impurity, and so on. But we must always accept some atomic arrangement that nature gives us. We haven't got anything, say, with a "checkerboard" arrangement, with the impurity atoms exactly arranged 1,000 angstroms apart, or in some other particular pattern. What could we do with layered structures with just the right layers? What would the properties of materials be if we could really arrange the atoms the way we want them? They would be very interesting to investigate theoretically. I can't see exactly what would happen, but I can hardly doubt that when we have some control of the arrangement of things on a small scale we will get an enormously greater range of possible properties that substances can have, and of different things that we can do. Consider, for example, a piece of material in which we make little coils and condensers (or their solid state analogs) 1,000 or 10,000 angstroms in a circuit, one right next to the other, over a large area, with little antennas sticking out at the other end-a whole series of circuits. Is it possible, for example, to emit light from a whole set of antennas, like we emit radio waves from an organized set of antennas to beam the radio programs to Europe? The same thing would be to beam the light out in a definite direction with very high intensity. (Perhaps such a beam is not very useful technically or economically.) I have thought about some of the problems of building electric circuits on a small scale, and the problem of resistance is serious. If you build a corresponding circuit on a small scale, its natural frequency goes up, since the wave length goes down as the scale; but the skin depth only decreases with the square root of the scale ratio, and so resistive problems are of increasing difficulty. Possibly we can beat resistance through the use of superconductivity if the frequency is not too high, or by other tricks. - Atoms in a small world - When we get to the very, very small world-say circuits of seven atoms-we

have a lot of new things that would happen that represent completely new opportunities for design. Atoms on a small scale behave like nothing on a large scale, for they satisfy the laws of quantum mechanics. So, as we go down and fiddle around with the atoms down there, we are working with different laws, and we can expect to do different things. We can manufacture in different ways. We can use, not just circuits, but some system involving the quantized energy levels, or the interactions of quantized spins, etc. Another thing we will notice is that, if we go down far enough, all of our devices can be mass produced so that they are absolutely perfect copies of one another. We cannot build two large machines so that the dimensions are exactly the same. But if your machine is only 100 atoms high, you only have to get it correct to one-half of one percent to make sure the other machine is exactly the same size-namely, 100 atoms high! At the atomic level, we have new kinds of forces and 7 new kinds of possibilities, new kinds of effects. The problems of manufacture and reproduction of materials will be quite different. I am, as I said, inspired by the biological phenomena in which chemical forces are used in repetitious fashion to produce all kinds of weird effects (one of which is the author). The principles of physics, as far as I can see, do not speak against the possibility of maneuvering things atom by atom. It is not an attempt to violate any laws; it is something, in principle, that can be done; but in practice, it has not been done because we are too big. Ultimately, we can do chemical synthesis. A chemist comes to us and says, "Look, I want a molecule that has the atoms arranged thus and so; make me that molecule." The chemist does a mysterious thing when he wants to make a molecule. He sees that it has got that ring, so he mixes this and that, and he shakes it, and he fiddles around. And, at the end of a difficult process, he usually does succeed in synthesizing what he wants. By the time I get my devices working, so that we can do it by physics, he will have figured out how to synthesize absolutely anything, so that this will really be useless. But it is interesting that it would be, in principle, possible (I think) for a physicist to synthesize any chemical substance that the chemist writes down. Give the orders and the physicist synthesizes it. How? Put the atoms down where the chemist says, and so you make the substance. The problems of chemistry and biology can be greatly helped if our ability to see what we are doing, and to do things on an atomic level, is ultimately developed- a development which I think cannot be avoided. Now, you might say, "Who

should do this and why should they do it?" Well, I pointed out a few of the economic applications, but I know that the reason that you would do it might be just for fun. But have some fun! Let's have a competition between laboratories. Let one laboratory make a tiny motor which it sends to another lab which sends it back with a thing that

Plate 7

fits inside the shaft of the first motor. - High school competition - Just for the fun of it, and in order to get kids interested in this field, I would propose that someone who has some contact with the high schools think of making some kind of high school competition. After all, we haven't even started in this field, and even the kids can write smaller than has ever been written before. They could have competition in high schools. The Los Angeles high school could send a pin to the Venice high school on which it says, "How's this?" They get the pin back, and in the dot of the "i" it says, "Not so hot." Perhaps this doesn't excite you to do it, and only economics will do so. Then I want to do something; but I can't do it at the present moment, because I haven't prepared the ground. It is my intention to offer a prize of 1,000 dollars to the first guy who can take the information on the page of a book and put it on an area 1/25,000 smaller in linear scale in such manner that it can be read by an electron microscope. And I want to offer another prize-if I can figure out how to phrase it so that I don't get into a mess of arguments about definitions-of another 1,000 dollars to the first guy who makes an operating electric motor-a rotating electric motor which can be controlled from the outside and, not counting the lead-in wires, is only 1/64 inch cube. I do not expect that such prizes will have to wait very long for claimants.

Computer Program 1 | "Text Split"

```
% ----------------------------------
% Splits a text string into groups of 1,536 characters
% [This code was run in Python]

import csv, math
import pandas as pd
from openpyxl import Workbook
import xlrd, xlwt
import xlsxwriter


def setup():
  global df, csv_df, var, list, break_var, char_list, byte, spot, char_var, char_list_2, char_list_3, char_list_4, total_list, plate_1_txt, plate_2_txt, plate_3_txt, plate_4_txt, leftover_txt #sets variables to be global so can be accessed in other methods
  df = open("Ascii binary combinations 2.csv") #opens up csv file in folder to be manipulated

  plate_1_txt = open("Plate1.txt", "w+") #creates new text files where text can be added to it
  plate_2_txt = open("Plate2.txt", "w+")
  plate_3_txt = open("Plate3.txt", "w+")
  plate_4_txt = open("Plate4.txt", "w+")
  leftover_txt = open("Remainders.txt", "w+")

  csv_df = csv.reader(df)

  total_list = [] #creating my lists where the split up text's characters will go
  char_list = []
  char_list_2 = []
  char_list_3 = []
  char_list_4 = []
  char_var = 0
  byte = 0
  spot = 0
  break_var = False
  var = input("Type Here -->") #asking for what the text you want split up is
  # print(var)
  list = [c for c in var] #makes the inputed text into a list of characters that can be manipulated easier
  # print(list)

  for i in range(len(list)+1):
    char_list.append("`")

  for u in range(len(list)+1):
    char_list_2.append("`")

  for v in range(len(list)+1):
    char_list_3.append("`")

  for w in range(len(list)+1):
    char_list_4.append("`")


def loop():
```

```
  global break_var, char_list, byte, spot, var_char, char_var, char_list_2, char_list_3, char_list_4,
plate_1_txt, plate_2_txt, plate_3_txt, plate_4_txt, leftover_txt

  for row in csv_df:  #loops through all characters in csv file
    for char in range(len(list)): #loops through all characters in inputed list
      if row[0] == list[char]: #splits up the text into the necessary list
        if math.floor((char % 384) / 192) == 0:
          if math.floor(char/4) % 2 == 0:
            char_list[char] = list[char]
          if math.floor(char/4) % 2 == 1:
            char_list_2[char] = list[char]
        if math.floor((char % 384) / 192) == 1:
          if math.floor(char/4) % 2 == 0:
            char_list_3[char] = list[char]
          if math.floor(char/4) % 2 == 1:
            char_list_4[char] = list[char]
        char_var += 1
      if char_var == 6144:
        break

  if char_var >= len(list): #taking each list and writing it into the new appropriate text file
    for i in range(len(char_list)):
      if i >= len(char_list):
        break_var = True
        break
      if char_list[i] == "`":
        char_list.pop(i)
        if i >= len(char_list):
          break
        while char_list[i] == "`":
          char_list.pop(i)
          if i >= len(char_list):
            break
      if i >= len(char_list):
        break_var = True
        break
      else:
        continue
    for i in range(len(char_list_2)): #repeat of previous code but for list 2
      if i >= len(char_list_2):
        break_var = True
        break
      if char_list_2[i] == "`":
        char_list_2.pop(i)
        if i >= len(char_list_2):
          break
        while char_list_2[i] == "`":
          char_list_2.pop(i)
          if i >= len(char_list_2):
            break
      if i >= len(char_list_2):
        break_var = True
        break
      else:
        continue
    for i in range(len(char_list_3)): #repeat of previous code but for list 3
```

```
        if i >= len(char_list_3):
            break_var = True
            break
        if char_list_3[i] == "`":
            char_list_3.pop(i)
            if i >= len(char_list_3):
                break
            while char_list_3[i] == "`":
                char_list_3.pop(i)
                if i >= len(char_list_3):
                    break
        if i >= len(char_list_3):
            break_var = True
            break
        else:
            continue
    for i in range(len(char_list_4)): #repeat of previous code but for list 4
        if i >= len(char_list_4):
            break_var = True
            break
        if char_list_4[i] == "`":
            char_list_4.pop(i)
            if i >= len(char_list_4):
                break
            while char_list_4[i] == "`":
                char_list_4.pop(i)
                if i >= len(char_list_4):
                    break
        if i >= len(char_list_4):
            break_var = True
            break
        else:
            continue
    for i in range(len(list)): #puts the left over character (the characters over the 6144 limit) into its own
text file
        if i >= len(list):
            break_var = True
            break
        if list[i] == "`":
            list.pop(i)
            if i >= len(list):
                break
            while list[i] == "`":
                list.pop(i)
                if i >= len(list):
                    break
        if i >= len(list):
            break_var = True
            break
        else:
            continue
    for i in range(len(char_list)):
        total_list.append(i)
    for i in range(len(char_list_2)):
        total_list.append(i)
    for i in range(len(char_list_3)):
```

```
            total_list.append(i)
        for i in range(len(char_list_4)):
            total_list.append(i)

        for i in range(len(total_list)):
            list.pop(0)

        for i in range(len(char_list)):
            plate_1_txt.write(char_list[i])
        for i in range(len(char_list_2)):
            plate_2_txt.write(char_list_2[i])
        for i in range(len(char_list_3)):
            plate_3_txt.write(char_list_3[i])
        for i in range(len(char_list_4)):
            plate_4_txt.write(char_list_4[i])
        for i in range(len(list)):
            leftover_txt.write(list[i])
        # for let in char_list:
        #     print(let)
        # for let_2 in char_list_2:
        #     print(let_2)
        # for let_3 in char_list_3:
        #     print(let_3)
        # for let_4 in char_list_4:
        #     print(let_4)
        # for lefts in list:
        #     print(lefts)
        plate_1_txt.close()
        plate_2_txt.close()
        plate_3_txt.close()
        plate_4_txt.close()
        leftover_txt.close()
        exit()



try:
    setup()
    while True:
        loop()

except Exception as reason:
    if len(reason.args)>0 and reason.args[0] == "User quit the game":
        print ("Crash.")

df.close()
```

Computer Program 2 | "Molbit Encoding"

```
% ----------------------------------
% Assigns oligopeptide molbits to input ASCII characters
% [This code was run in Python]
import csv, math
import pandas as pd
from openpyxl import Workbook
import xlrd, xlwt
import xlsxwriter

#Creates all variables and sets up the code to be prepared to loop
def setup():
  global df, csv_df, var, list, break_var, char_list, byte, spot, char_var, i, vol, source_well_row, source_well_spot, source_well_section, message_length, char_counter #makes variables global so they can be used in other methods
  global dest_plate, dest_well_spot, dest_well_section, vol_1, vol_2, vol_3, vol_4, vol_5, vol_6, vol_7, vol_8, vol_9, vol_10, vol_11, vol_12, vol_13, vol_14
  global vol_15, vol_16, vol_17, vol_18, vol_19, vol_20, vol_21, vol_22, vol_23, vol_24, vol_25, vol_26, vol_27, vol_28, vol_29, vol_30, vol_31, vol_32
  global vol_list, vol_max, total_list, ws, wb, last_section, last_spot, df_2, csv_df_2, volume_list

  df_2 = open("PeptideVolumes.csv") #opens up csv files in folder to be manipulated in code
  df = open("Ascii binary combinations.csv")
  csv_df_2 = csv.reader(df_2)
  csv_df = csv.reader(df)

  volume_list = [] #making a list of the specific volumes for each peptide
  for row in csv_df_2:
    volume_list.append(float(row[0]))

  last_spot = False #Defining all variables
  last_section = False
  wb = xlsxwriter.Workbook('384.xlsx')
  ws = wb.add_worksheet("New Sheet")
  vol_1 = 0
  vol_2 = 0
  vol_3 = 0
  vol_4 = 0
  vol_5 = 0
  vol_6 = 0
  vol_7 = 0
  vol_8 = 0
  vol_9 = 0
  vol_10 = 0
  vol_11 = 0
  vol_12 = 0
  vol_13 = 0
  vol_14 = 0
  vol_15 = 0
  vol_16 = 0
  vol_17 = 0
  vol_18 = 0
  vol_19 = 0
  vol_20 = 0
  vol_21 = 0
```

```
  vol_22 = 0
  vol_23 = 0
  vol_24 = 0
  vol_25 = 0
  vol_26 = 0
  vol_27 = 0
  vol_28 = 0
  vol_29 = 0
  vol_30 = 0
  vol_31 = 0
  vol_32 = 0
  i = 0
  vol = 0
  total_list = []
  vol_list = []
  char_counter = 0
  dest_well_section = "A"
  dest_well_spot = "01"
  dest_plate = 1
  char_list = []
  source_well_row = 1
  source_well_spot = 1
  source_well_section = 2
  char_var = 0
  byte = 0
  spot = 0
  break_var = False

  var = input("Type Here -->") #asking for the characters that are being encoded
  vol_max = int(input("Volume Max?")) #asking for the maximum volume in each destination well
  dest_plate = int(input("Destination Plate?")) #asking which destination plate it is being transfered to
  temp = input("Hit 'Enter'")
  list = [c for c in var] #splitting up the input into a list of each seperate character
  message_length = len(list)

  while i < len(list)*7:
     char_list.append("`")
     i += 1


def loop():
  global break_var, char_list, byte, spot, char_var, i, vol, source_well_row, source_well_spot,
source_well_section, message_length, char_counter
  global dest_plate, dest_well_spot, dest_well_section, vol_1, vol_2, vol_3, vol_4, vol_5, vol_6, vol_7,
vol_8, vol_9, vol_10, vol_11, vol_12, vol_13, vol_14
  global vol_15, vol_16, vol_17, vol_18, vol_19, vol_20, vol_21, vol_22, vol_23, vol_24, vol_25, vol_26,
vol_27, vol_28, vol_29, vol_30, vol_31, vol_32
  global vol_list, total_list, ws, wb, last_section, last_spot
  for row in csv_df: #looping through all the characters in the csv file
     for char in range(len(list)):#looping through each character in the char list
        if row[0] == list[char]: #checking to see if the current char in the list matches one of the
characters in the csv file

           char_spot = char * 7 #setting up initial variables for loop
           temp_list = [c for c in row[1]]
           byte = (char % 4) + 1
```

```
spot = math.floor(char / 4) + 1
temp_sect = math.floor(char/64)

dest_well_spot = "13" #calculating the destination well row
dest_well_section = "A"
if 25 <= math.floor(char / 4) + 1 < 49:
    dest_well_section = "B"
if 49 <= math.floor(char / 4) + 1 < 73:
    dest_well_section = "C"
if 73 <= math.floor(char / 4) + 1 < 97:
    dest_well_section = "D"
if 97 <= math.floor(char / 4) + 1 < 121:
    dest_well_section = "E"
if 121 <= math.floor(char / 4) + 1 < 145:
    dest_well_section = "F"
if 145 <= math.floor(char / 4) + 1 < 169:
    dest_well_section = "G"
if 169 <= math.floor(char / 4) + 1 < 193:
    dest_well_section = "H"
if 193 <= math.floor(char / 4) + 1 < 217:
    dest_well_section = "I"
if 217 <= math.floor(char / 4) + 1 < 241:
    dest_well_section = "J"
if 241 <= math.floor(char / 4) + 1 < 265:
    dest_well_section = "K"
if 265 <= math.floor(char / 4) + 1 < 289:
    dest_well_section = "L"
if 289 <= math.floor(char / 4) + 1 < 313:
    dest_well_section = "M"
if 313 <= math.floor(char / 4) + 1 < 337:
    dest_well_section = "N"
if 337 <= math.floor(char / 4) + 1 < 361:
    dest_well_section = "O"
if 361 <= math.floor(char / 4) + 1 < 385:
    dest_well_section = "P"
    last_section = True
if math.floor(char/4) % 24+1 < 10:
    dest_well_spot = "0{0}".format((math.floor(char / 4) %24) + 1)
else:
    dest_well_spot = (math.floor(char / 4) %24) +1

if temp_list[1] == "1": #Calculating what the destination column number will be
    temp_well = ((char*8)+2)%32
    # print(temp_well)
    if temp_well == 2:
        vol_2 += volume_list[1]
        vol = volume_list[1]
        source_well_section = "A"
        if vol_2 < vol_max:
            source_well_spot = "13"
        if vol_max <= vol_2 < vol_max*2:
            source_well_spot = "14"
        if vol_max*2 <= vol_2 < vol_max*3:
            source_well_spot = "15"
        if vol_max*3 <= vol_2 < vol_max*4:
            source_well_spot = "16"
```

```
    if vol_max*4 <= vol_2 < vol_max*5:
        source_well_spot = "17"
    if vol_max*5 <= vol_2 < vol_max*6:
        source_well_spot = "18"
    if vol_max*6 <= vol_2 < vol_max*7:
        source_well_spot = "19"
    if vol_max*7 <= vol_2 < vol_max*8:
        source_well_spot = "20"
    if vol_max*8 <= vol_2 < vol_max*9:
        source_well_spot = "21"
    if vol_max*9<= vol_2 < vol_max*10:
        source_well_spot = "22"
    if vol_max*10 <= vol_2 < vol_max*11:
        source_well_spot = "23"
    if vol_2 >= vol_max*11:
        source_well_spot = "24"
if temp_well == 10:
    vol_10 += volume_list[9]
    vol = volume_list[9]
    source_well_section = "E"
    if vol_10 < vol_max:
        source_well_spot = "13"
    if vol_max<= vol_10 < vol_max*2:
        source_well_spot = "14"
    if vol_max*2 <= vol_10 < vol_max*3:
        source_well_spot = "15"
    if vol_max*3 <= vol_10 < vol_max*4:
        source_well_spot = "16"
    if vol_max*4 <= vol_10 < vol_max*5:
        source_well_spot = "17"
    if vol_max*5 <= vol_10 < vol_max*6:
        source_well_spot = "18"
    if vol_max*6 <= vol_10 < vol_max*7:
        source_well_spot = "19"
    if vol_max*7 <= vol_10 < vol_max*8:
        source_well_spot = "20"
    if vol_max*8 <= vol_10 < vol_max*9:
        source_well_spot = "21"
    if vol_max*9 <= vol_10 < vol_max*10:
        source_well_spot = "22"
    if vol_max*10 <= vol_10 < vol_max*11:
        source_well_spot = "23"
    if vol_10 >= vol_max*11:
        source_well_spot = "24"
if temp_well == 18:
    vol_18 += volume_list[17]
    vol = volume_list[17]
    source_well_section = "I"
    if vol_18 < vol_max:
        source_well_spot = "13"
    if vol_max<= vol_18 < vol_max*2:
        source_well_spot = "14"
    if vol_max*2 <= vol_18 < vol_max*3:
        source_well_spot = "15"
    if vol_max*3 <= vol_18 < vol_max*4:
        source_well_spot = "16"
```

```
                if vol_max*4 <= vol_18 < vol_max*5:
                    source_well_spot = "17"
                if vol_max*5 <= vol_18 < vol_max*6:
                    source_well_spot = "18"
                if vol_max*6 <= vol_18 < vol_max*7:
                    source_well_spot = "19"
                if vol_max*7 <= vol_18 < vol_max*8:
                    source_well_spot = "20"
                if vol_max*8 <= vol_18 < vol_max*9:
                    source_well_spot = "21"
                if vol_max*9 <= vol_18 < vol_max*10:
                    source_well_spot = "22"
                if vol_max*10 <= vol_18 < vol_max*11:
                    source_well_spot = "23"
                if vol_18 >= vol_max*11:
                    source_well_spot = "24"
            if temp_well == 26:
                vol_26 += volume_list[25]
                vol = volume_list[25]
                source_well_section = "M"
                if vol_26 < vol_max:
                    source_well_spot = "13"
                if vol_max<= vol_26 < vol_max*2:
                    source_well_spot = "14"
                if vol_max*2 <= vol_26 < vol_max*3:
                    source_well_spot = "15"
                if vol_max*3 <= vol_26 < vol_max*4:
                    source_well_spot = "16"
                if vol_max*4 <= vol_26 < vol_max*5:
                    source_well_spot = "17"
                if vol_max*5 <= vol_26 < vol_max*6:
                    source_well_spot = "18"
                if vol_max*6 <= vol_26 < vol_max*7:
                    source_well_spot = "19"
                if vol_max*7 <= vol_26 < vol_max*8:
                    source_well_spot = "20"
                if vol_max*8 <= vol_26 < vol_max*9:
                    source_well_spot = "21"
                if vol_max*9 <= vol_26 < vol_max*10:
                    source_well_spot = "22"
                if vol_max*10 <= vol_26 < vol_max*11:
                    source_well_spot = "23"
                if vol_26 >= vol_max*11:
                    source_well_spot = "24"
            temp_char = "Master {0}{1} {2} {3}{4} {5}".format(source_well_section,
source_well_spot, dest_plate, dest_well_section, dest_well_spot, vol) #adding to the final list what the
input table row should look like for one character
            char_list[char_spot] = temp_char

        if temp_list[2] == "1": #repeated for all the same kind of code in the rest of this loop
            temp_well = ((char * 8) + 3)%32
            source_well_spot = "01"
            if temp_well == 3:
                vol_3 += volume_list[2]
                vol = volume_list[2]
                source_well_section = "B"
```

```
    if vol_3 < vol_max:
        source_well_spot = "01"
    if vol_max<= vol_3 < vol_max*2:
        source_well_spot = "02"
    if vol_max*2 <= vol_3 < vol_max*3:
        source_well_spot = "03"
    if vol_max*3 <= vol_3 < vol_max*4:
        source_well_spot = "04"
    if vol_max*4 <= vol_3 < vol_max*5:
        source_well_spot = "05"
    if vol_max*5 <= vol_3 < vol_max*6:
        source_well_spot = "06"
    if vol_max*6 <= vol_3 < vol_max*7:
        source_well_spot = "07"
    if vol_max*7 <= vol_3 < vol_max*8:
        source_well_spot = "08"
    if vol_max*8 <= vol_3 < vol_max*9:
        source_well_spot = "09"
    if vol_max*9 <= vol_3 < vol_max*10:
        source_well_spot = "10"
    if vol_max*10 <= vol_3 < vol_max*11:
        source_well_spot = "11"
    if vol_3 >= vol_max*11:
        source_well_spot = "12"
if temp_well == 11:
    vol_11 += volume_list[10]
    vol = volume_list[10]
    source_well_section = "F"
    if vol_11 < vol_max:
        source_well_spot = "01"
    if vol_max<= vol_11 < vol_max*2:
        source_well_spot = "02"
    if vol_max*2 <= vol_11 < vol_max*3:
        source_well_spot = "03"
    if vol_max*3 <= vol_11 < vol_max*4:
        source_well_spot = "04"
    if vol_max*4 <= vol_11 < vol_max*5:
        source_well_spot = "05"
    if vol_max*5 <= vol_11 < vol_max*6:
        source_well_spot = "06"
    if vol_max*6 <= vol_11 < vol_max*7:
        source_well_spot = "07"
    if vol_max*7 <= vol_11 < vol_max*8:
        source_well_spot = "08"
    if vol_max*8 <= vol_11 < vol_max*9:
        source_well_spot = "09"
    if vol_max*9 <= vol_11 < vol_max*10:
        source_well_spot = "10"
    if vol_max*10 <= vol_11 < vol_max*11:
        source_well_spot = "11"
    if vol_11 >= vol_max*11:
        source_well_spot = "12"
if temp_well == 19:
    vol_19 += volume_list[18]
    vol = volume_list[18]
    source_well_section = "J"
```

```
                if vol_19 < vol_max:
                    source_well_spot = "01"
                if vol_max<= vol_19 < vol_max*2:
                    source_well_spot = "02"
                if vol_max*2 <= vol_19 < vol_max*3:
                    source_well_spot = "03"
                if vol_max*3 <= vol_19 < vol_max*4:
                    source_well_spot = "04"
                if vol_max*4 <= vol_19 < vol_max*5:
                    source_well_spot = "05"
                if vol_max*5 <= vol_19 < vol_max*6:
                    source_well_spot = "06"
                if vol_max*6 <= vol_19 < vol_max*7:
                    source_well_spot = "07"
                if vol_max*7 <= vol_19 < vol_max*8:
                    source_well_spot = "08"
                if vol_max*8 <= vol_19 < vol_max*9:
                    source_well_spot = "09"
                if vol_max*9 <= vol_19 < vol_max*10:
                    source_well_spot = "10"
                if vol_max*10 <= vol_19 < vol_max*11:
                    source_well_spot = "11"
                if vol_19 >= vol_max*11:
                    source_well_spot = "12"
            if temp_well == 27:
                vol_27 += volume_list[26]
                vol = volume_list[26]
                source_well_section = "N"
                if vol_27 < vol_max:
                    source_well_spot = "01"
                if vol_max<= vol_27 < vol_max*2:
                    source_well_spot = "02"
                if vol_max*2 <= vol_27 < vol_max*3:
                    source_well_spot = "03"
                if vol_max*3 <= vol_27 < vol_max*4:
                    source_well_spot = "04"
                if vol_max*4 <= vol_27 < vol_max*5:
                    source_well_spot = "05"
                if vol_max*5 <= vol_27 < vol_max*6:
                    source_well_spot = "06"
                if vol_max*6 <= vol_27 < vol_max*7:
                    source_well_spot = "07"
                if vol_max*7 <= vol_27 < vol_max*8:
                    source_well_spot = "08"
                if vol_max*8 <= vol_27 < vol_max*9:
                    source_well_spot = "09"
                if vol_max*9 <= vol_27 < vol_max*10:
                    source_well_spot = "10"
                if vol_max*10 <= vol_27 < vol_max*11:
                    source_well_spot = "11"
                if vol_27 >= vol_max*11:
                    source_well_spot = "12"
            temp_char = "Master {0}{1} {2} {3}{4} {5}".format(source_well_section,
source_well_spot, dest_plate, dest_well_section, dest_well_spot, vol)
            char_list[char_spot+1] = temp_char
        if temp_list[3] == "1":
```

```
temp_well = ((char * 8) + 4)%32
source_well_spot = "13"
if temp_well == 4:
    vol_4 += volume_list[3]
    vol = volume_list[3]
    source_well_section = "B"
    if vol_4 < vol_max:
        source_well_spot = "13"
    if vol_max<= vol_4 < vol_max*2:
        source_well_spot = "14"
    if vol_max*2 <= vol_4 < vol_max*3:
        source_well_spot = "15"
    if vol_max*3 <= vol_4 < vol_max*4:
        source_well_spot = "16"
    if vol_max*4 <= vol_4 < vol_max*5:
        source_well_spot = "17"
    if vol_max*5 <= vol_4 < vol_max*6:
        source_well_spot = "18"
    if vol_max*6 <= vol_4 < vol_max*7:
        source_well_spot = "19"
    if vol_max*7 <= vol_4 < vol_max*8:
        source_well_spot = "20"
    if vol_max*8 <= vol_4 < vol_max*9:
        source_well_spot = "21"
    if vol_max*9 <= vol_4 < vol_max*10:
        source_well_spot = "22"
    if vol_max*10 <= vol_4 < vol_max*11:
        source_well_spot = "23"
    if vol_4 >= vol_max*11:
        source_well_spot = "24"
if temp_well == 12:
    vol_12 += volume_list[11]
    vol = volume_list[11]
    source_well_section = "F"
    if vol_12 < vol_max:
        source_well_spot = "13"
    if vol_max<= vol_12 < vol_max*2:
        source_well_spot = "14"
    if vol_max*2 <= vol_12 < vol_max*3:
        source_well_spot = "15"
    if vol_max*3 <= vol_12 < vol_max*4:
        source_well_spot = "16"
    if vol_max*4 <= vol_12 < vol_max*5:
        source_well_spot = "17"
    if vol_max*5 <= vol_12 < vol_max*6:
        source_well_spot = "18"
    if vol_max*6 <= vol_12 < vol_max*7:
        source_well_spot = "19"
    if vol_max*7 <= vol_12 < vol_max*8:
        source_well_spot = "20"
    if vol_max*8 <= vol_12 < vol_max*9:
        source_well_spot = "21"
    if vol_max*9 <= vol_12 < vol_max*10:
        source_well_spot = "22"
    if vol_max*10 <= vol_12 < vol_max*11:
        source_well_spot = "23"
```

```
    if vol_12 >= vol_max*11:
        source_well_spot = "24"
if temp_well == 20:
    vol_20 += volume_list[19]
    vol = volume_list[19]
    source_well_section = "J"
    if vol_20 < vol_max:
        source_well_spot = "13"
    if vol_max<= vol_20 < vol_max*2:
        source_well_spot = "14"
    if vol_max*2 <= vol_20 < vol_max*3:
        source_well_spot = "15"
    if vol_max*3 <= vol_20 < vol_max*4:
        source_well_spot = "16"
    if vol_max*4 <= vol_20 < vol_max*5:
        source_well_spot = "17"
    if vol_max*5 <= vol_20 < vol_max*6:
        source_well_spot = "18"
    if vol_max*6 <= vol_20 < vol_max*7:
        source_well_spot = "19"
    if vol_max*7 <= vol_20 < vol_max*8:
        source_well_spot = "20"
    if vol_max*8 <= vol_20 < vol_max*9:
        source_well_spot = "21"
    if vol_max*9 <= vol_20 < vol_max*10:
        source_well_spot = "22"
    if vol_max*10 <= vol_20 < vol_max*11:
        source_well_spot = "23"
    if vol_20 >= vol_max*11:
        source_well_spot = "24"
if temp_well == 28:
    vol_28 += volume_list[27]
    vol = volume_list[27]
    source_well_section = "N"
    if vol_28 < vol_max:
        source_well_spot = "13"
    if vol_max<= vol_28 < vol_max*2:
        source_well_spot = "14"
    if vol_max*2 <= vol_28 < vol_max*3:
        source_well_spot = "15"
    if vol_max*3 <= vol_28 < vol_max*4:
        source_well_spot = "16"
    if vol_max*4 <= vol_28 < vol_max*5:
        source_well_spot = "17"
    if vol_max*5 <= vol_28 < vol_max*6:
        source_well_spot = "18"
    if vol_max*6 <= vol_28 < vol_max*7:
        source_well_spot = "19"
    if vol_max*7 <= vol_28 < vol_max*8:
        source_well_spot = "20"
    if vol_max*8 <= vol_28 < vol_max*9:
        source_well_spot = "21"
    if vol_max*9 <= vol_28 < vol_max*10:
        source_well_spot = "22"
    if vol_max*10 <= vol_28 < vol_max*11:
        source_well_spot = "23"
```

```
                if vol_28 >= vol_max*11:
                    source_well_spot = "24"
            temp_char = "Master {0}{1} {2} {3}{4} {5}".format(source_well_section,
source_well_spot, dest_plate, dest_well_section, dest_well_spot, vol, list[char])
            char_list[char_spot+2] = temp_char
        if temp_list[4] == "1":
            temp_well = ((char * 8) + 5)%32
            source_well_spot = "01"
            if temp_well == 5:
                vol_5 += volume_list[4]
                vol = volume_list[4]
                source_well_section = "C"
                if vol_5 < vol_max:
                    source_well_spot = "01"
                if vol_max<= vol_5 < vol_max*2:
                    source_well_spot = "02"
                if vol_max*2 <= vol_5 < vol_max*3:
                    source_well_spot = "03"
                if vol_max*3 <= vol_5 < vol_max*4:
                    source_well_spot = "04"
                if vol_max*4 <= vol_5 < vol_max*5:
                    source_well_spot = "05"
                if vol_max*5 <= vol_5 < vol_max*6:
                    source_well_spot = "06"
                if vol_max*6 <= vol_5 < vol_max*7:
                    source_well_spot = "07"
                if vol_max*7 <= vol_5 < vol_max*8:
                    source_well_spot = "08"
                if vol_max*8 <= vol_5 < vol_max*9:
                    source_well_spot = "09"
                if vol_max*9 <= vol_5 < vol_max*10:
                    source_well_spot = "10"
                if vol_max*10 <= vol_5 < vol_max*11:
                    source_well_spot = "11"
                if vol_5 >= vol_max*11:
                    source_well_spot = "12"
            if temp_well == 13:
                vol_13 += volume_list[12]
                vol = volume_list[12]
                source_well_section = "G"
                if vol_13 < vol_max:
                    source_well_spot = "01"
                if vol_max<= vol_13 < vol_max*2:
                    source_well_spot = "02"
                if vol_max*2 <= vol_13 < vol_max*3:
                    source_well_spot = "03"
                if vol_max*3 <= vol_13 < vol_max*4:
                    source_well_spot = "04"
                if vol_max*4 <= vol_13 < vol_max*5:
                    source_well_spot = "05"
                if vol_max*5 <= vol_13 < vol_max*6:
                    source_well_spot = "06"
                if vol_max*6 <= vol_13 < vol_max*7:
                    source_well_spot = "07"
                if vol_max*7 <= vol_13 < vol_max*8:
                    source_well_spot = "08"
```

```
    if vol_max*8 <= vol_13 < vol_max*9:
        source_well_spot = "09"
    if vol_max*9 <= vol_13 < vol_max*10:
        source_well_spot = "10"
    if vol_max*10 <= vol_13 < vol_max*11:
        source_well_spot = "11"
    if vol_13 >= vol_max*11:
        source_well_spot = "12"
if temp_well == 21:
    vol_21 += volume_list[20]
    vol = volume_list[20]
    source_well_section = "K"
    if vol_21 < vol_max:
        source_well_spot = "01"
    if vol_max<= vol_21 < vol_max*2:
        source_well_spot = "02"
    if vol_max*2 <= vol_21 < vol_max*3:
        source_well_spot = "03"
    if vol_max*3 <= vol_21 < vol_max*4:
        source_well_spot = "04"
    if vol_max*4 <= vol_21 < vol_max*5:
        source_well_spot = "05"
    if vol_max*5 <= vol_21 < vol_max*6:
        source_well_spot = "06"
    if vol_max*6 <= vol_21 < vol_max*7:
        source_well_spot = "07"
    if vol_max*7 <= vol_21 < vol_max*8:
        source_well_spot = "08"
    if vol_max*8 <= vol_21 < vol_max*9:
        source_well_spot = "09"
    if vol_max*9 <= vol_21 < vol_max*10:
        source_well_spot = "10"
    if vol_max*10 <= vol_21 < vol_max*11:
        source_well_spot = "11"
    if vol_21 >= vol_max*11:
        source_well_spot = "12"
if temp_well == 29:
    vol_29 += volume_list[28]
    vol = volume_list[28]
    source_well_section = "O"
    if vol_29 < vol_max:
        source_well_spot = "01"
    if vol_max<= vol_29 < vol_max*2:
        source_well_spot = "02"
    if vol_max*2 <= vol_29 < vol_max*3:
        source_well_spot = "03"
    if vol_max*3 <= vol_29 < vol_max*4:
        source_well_spot = "04"
    if vol_max*4 <= vol_29 < vol_max*5:
        source_well_spot = "05"
    if vol_max*5 <= vol_29 < vol_max*6:
        source_well_spot = "06"
    if vol_max*6 <= vol_29 < vol_max*7:
        source_well_spot = "07"
    if vol_max*7 <= vol_29 < vol_max*8:
        source_well_spot = "08"
```

```
                if vol_max*8 <= vol_29 < vol_max*9:
                    source_well_spot = "09"
                if vol_max*9 <= vol_29 < vol_max*10:
                    source_well_spot = "10"
                if vol_max*10 <= vol_29 < vol_max*11:
                    source_well_spot = "11"
                if vol_29 >= vol_max*11:
                    source_well_spot = "12"
            temp_char = "Master {0}{1} {2} {3}{4} {5}".format(source_well_section,
source_well_spot, dest_plate, dest_well_section, dest_well_spot, vol)
            char_list[char_spot+3] = temp_char
        if temp_list[5] == "1":
            temp_well = ((char * 8) + 6)%32
            source_well_spot = "13"
            if temp_well == 6:
                vol_6 += volume_list[5]
                vol = volume_list[5]
                source_well_section = "C"
                if vol_6 < vol_max:
                    source_well_spot = "13"
                if vol_max<= vol_6 < vol_max*2:
                    source_well_spot = "14"
                if vol_max*2 <= vol_6 < vol_max*3:
                    source_well_spot = "15"
                if vol_max*3 <= vol_6 < vol_max*4:
                    source_well_spot = "16"
                if vol_max*4 <= vol_6 < vol_max*5:
                    source_well_spot = "17"
                if vol_max*5 <= vol_6 < vol_max*6:
                    source_well_spot = "18"
                if vol_max*6 <= vol_6 < vol_max*7:
                    source_well_spot = "19"
                if vol_max*7 <= vol_6 < vol_max*8:
                    source_well_spot = "20"
                if vol_max*8 <= vol_6 < vol_max*9:
                    source_well_spot = "21"
                if vol_max*9 <= vol_6 < vol_max*10:
                    source_well_spot = "22"
                if vol_max*10 <= vol_6 < vol_max*11:
                    source_well_spot = "23"
                if vol_6 >= vol_max*11:
                    source_well_spot = "24"
            if temp_well == 14:
                vol_14 += volume_list[13]
                vol = volume_list[13]
                source_well_section = "G"
                if vol_14 < vol_max:
                    source_well_spot = "13"
                if vol_max<= vol_14 < vol_max*2:
                    source_well_spot = "14"
                if vol_max*2 <= vol_14 < vol_max*3:
                    source_well_spot = "15"
                if vol_max*3 <= vol_14 < vol_max*4:
                    source_well_spot = "16"
                if vol_max*4 <= vol_14 < vol_max*5:
                    source_well_spot = "17"
```

```
    if vol_max*5 <= vol_14 < vol_max*6:
        source_well_spot = "18"
    if vol_max*6 <= vol_14 < vol_max*7:
        source_well_spot = "19"
    if vol_max*7 <= vol_14 < vol_max*8:
        source_well_spot = "20"
    if vol_max*8 <= vol_14 < vol_max*9:
        source_well_spot = "21"
    if vol_max*9 <= vol_14 < vol_max*10:
        source_well_spot = "22"
    if vol_max*10 <= vol_14 < vol_max*11:
        source_well_spot = "23"
    if vol_14 >= vol_max*11:
        source_well_spot = "24"
if temp_well == 22:
    vol_22 += volume_list[21]
    vol = volume_list[21]
    source_well_section = "K"
    if vol_22 < vol_max:
        source_well_spot = "13"
    if vol_max<= vol_22 < vol_max*2:
        source_well_spot = "14"
    if vol_max*2 <= vol_22 < vol_max*3:
        source_well_spot = "15"
    if vol_max*3 <= vol_22 < vol_max*4:
        source_well_spot = "16"
    if vol_max*4 <= vol_22 < vol_max*5:
        source_well_spot = "17"
    if vol_max*5 <= vol_22 < vol_max*6:
        source_well_spot = "18"
    if vol_max*6 <= vol_22 < vol_max*7:
        source_well_spot = "19"
    if vol_max*7 <= vol_22 < vol_max*8:
        source_well_spot = "20"
    if vol_max*8 <= vol_22 < vol_max*9:
        source_well_spot = "21"
    if vol_max*9 <= vol_22 < vol_max*10:
        source_well_spot = "22"
    if vol_max*10 <= vol_22 < vol_max*11:
        source_well_spot = "23"
    if vol_22 >= vol_max*11:
        source_well_spot = "24"
if temp_well == 30:
    vol_30 += volume_list[29]
    vol = volume_list[29]
    source_well_section = "O"
    if vol_30 < vol_max:
        source_well_spot = "13"
    if vol_max<= vol_30 < vol_max*2:
        source_well_spot = "14"
    if vol_max*2 <= vol_30 < vol_max*3:
        source_well_spot = "15"
    if vol_max*3 <= vol_30 < vol_max*4:
        source_well_spot = "16"
    if vol_max*4 <= vol_30 < vol_max*5:
        source_well_spot = "17"
```

```
                if vol_max*5 <= vol_30< vol_max*6:
                    source_well_spot = "18"
                if vol_max*6 <= vol_30 < vol_max*7:
                    source_well_spot = "19"
                if vol_max*7 <= vol_30 < vol_max*8:
                    source_well_spot = "20"
                if vol_max*8 <= vol_30 < vol_max*9:
                    source_well_spot = "21"
                if vol_max*9 <= vol_30 < vol_max*10:
                    source_well_spot = "22"
                if vol_max*10 <= vol_30 < vol_max*11:
                    source_well_spot = "23"
                if vol_30 >= vol_max*11:
                    source_well_spot = "24"
            temp_char = "Master {0}{1} {2} {3}{4} {5}".format(source_well_section,
source_well_spot, dest_plate, dest_well_section, dest_well_spot, vol)
            char_list[char_spot+4] = temp_char
        if temp_list[6] == "1":
            temp_well = ((char * 8) + 7)%32
            source_well_spot = "01"
            if temp_well == 7:
                vol_7 += volume_list[6]
                vol = volume_list[6]
                source_well_section = "D"
                if vol_7 < vol_max:
                    source_well_spot = "01"
                if vol_max<= vol_7 < vol_max*2:
                    source_well_spot = "02"
                if vol_max*2 <= vol_7 < vol_max*3:
                    source_well_spot = "03"
                if vol_max*3 <= vol_7 < vol_max*4:
                    source_well_spot = "04"
                if vol_max*4 <= vol_7 < vol_max*5:
                    source_well_spot = "05"
                if vol_max*5 <= vol_7 < vol_max*6:
                    source_well_spot = "06"
                if vol_max*6 <= vol_7 < vol_max*7:
                    source_well_spot = "07"
                if vol_max*7 <= vol_7 < vol_max*8:
                    source_well_spot = "08"
                if vol_max*8 <= vol_7 < vol_max*9:
                    source_well_spot = "09"
                if vol_max*9 <= vol_7 < vol_max*10:
                    source_well_spot = "10"
                if vol_max*10 <= vol_7 < vol_max*11:
                    source_well_spot = "11"
                if vol_7 >= vol_max*11:
                    source_well_spot = "12"
            if temp_well == 15:
                vol_15 += volume_list[14]
                vol = volume_list[14]
                source_well_section = "H"
                if vol_15 < vol_max:
                    source_well_spot = "01"
                if vol_max<= vol_15 < vol_max*2:
                    source_well_spot = "02"
```

```
    if vol_max*2 <= vol_15 < vol_max*3:
        source_well_spot = "03"
    if vol_max*3 <= vol_15 < vol_max*4:
        source_well_spot = "04"
    if vol_max*4 <= vol_15 < vol_max*5:
        source_well_spot = "05"
    if vol_max*5 <= vol_15 < vol_max*6:
        source_well_spot = "06"
    if vol_max*6 <= vol_15 < vol_max*7:
        source_well_spot = "07"
    if vol_max*7 <= vol_15 < vol_max*8:
        source_well_spot = "08"
    if vol_max*8 <= vol_15 < vol_max*9:
        source_well_spot = "09"
    if vol_max*9 <= vol_15 < vol_max*10:
        source_well_spot = "10"
    if vol_max*10 <= vol_15 < vol_max*11:
        source_well_spot = "11"
    if vol_15 >= vol_max*11:
        source_well_spot = "12"
if temp_well == 23:
    vol_23 += volume_list[22]
    vol = volume_list[22]
    source_well_section = "L"
    if vol_23 < vol_max:
        source_well_spot = "01"
    if vol_max<= vol_23 < vol_max*2:
        source_well_spot = "02"
    if vol_max*2 <= vol_23 < vol_max*3:
        source_well_spot = "03"
    if vol_max*3 <= vol_23 < vol_max*4:
        source_well_spot = "04"
    if vol_max*4 <= vol_23 < vol_max*5:
        source_well_spot = "05"
    if vol_max*5 <= vol_23 < vol_max*6:
        source_well_spot = "06"
    if vol_max*6 <= vol_23 < vol_max*7:
        source_well_spot = "07"
    if vol_max*7 <= vol_23 < vol_max*8:
        source_well_spot = "08"
    if vol_max*8 <= vol_23 < vol_max*9:
        source_well_spot = "09"
    if vol_max*9 <= vol_23 < vol_max*10:
        source_well_spot = "10"
    if vol_max*10 <= vol_23 < vol_max*11:
        source_well_spot = "11"
    if vol_23 >= vol_max*11:
        source_well_spot = "12"
if temp_well == 31:
    vol_31 += volume_list[30]
    vol = volume_list[30]
    source_well_section = "P"
    if vol_31 < vol_max:
        source_well_spot = "01"
    if vol_max<= vol_31 < vol_max*2:
        source_well_spot = "02"
```

```
                if vol_max*2 <= vol_31 < vol_max*3:
                    source_well_spot = "03"
                if vol_max*3 <= vol_31 < vol_max*4:
                    source_well_spot = "04"
                if vol_max*4 <= vol_31 < vol_max*5:
                    source_well_spot = "05"
                if vol_max*5 <= vol_31 < vol_max*6:
                    source_well_spot = "06"
                if vol_max*6 <= vol_31 < vol_max*7:
                    source_well_spot = "07"
                if vol_max*7 <= vol_31 < vol_max*8:
                    source_well_spot = "08"
                if vol_max*8 <= vol_31 < vol_max*9:
                    source_well_spot = "09"
                if vol_max*9 <= vol_31 < vol_max*10:
                    source_well_spot = "10"
                if vol_max*10 <= vol_31 < vol_max*11:
                    source_well_spot = "11"
                if vol_31 >= vol_max*11:
                    source_well_spot = "12"
            temp_char = "Master {0}{1} {2} {3}{4} {5}".format(source_well_section,
source_well_spot, dest_plate, dest_well_section, dest_well_spot, vol)
            char_list[char_spot+5] = temp_char
        if temp_list[7] == "1":
            temp_well = ((char * 8) + 8)%32
            if temp_well == 0:
                temp_well = 32
            source_well_spot = "13"
            if temp_well == 8:
                vol_8 += volume_list[7]
                vol = volume_list[7]
                source_well_section = "D"
                if vol_8 < vol_max:
                    source_well_spot = "13"
                if vol_max<= vol_8 < vol_max*2:
                    source_well_spot = "14"
                if vol_max*2 <= vol_8 < vol_max*3:
                    source_well_spot = "15"
                if vol_max*3 <= vol_8 < vol_max*4:
                    source_well_spot = "16"
                if vol_max*4 <= vol_8 < vol_max*5:
                    source_well_spot = "17"
                if vol_max*5 <= vol_8 < vol_max*6:
                    source_well_spot = "18"
                if vol_max*6 <= vol_8 < vol_max*7:
                    source_well_spot = "19"
                if vol_max*7 <= vol_8 < vol_max*8:
                    source_well_spot = "20"
                if vol_max*8 <= vol_8 < vol_max*9:
                    source_well_spot = "21"
                if vol_max*9 <= vol_8 < vol_max*10:
                    source_well_spot = "22"
                if vol_max*10 <= vol_8 < vol_max*11:
                    source_well_spot = "23"
                if vol_8 >= vol_max*11:
                    source_well_spot = "24"
```

```
if temp_well == 16:
    vol_16 += volume_list[15]
    vol = volume_list[15]
    source_well_section = "H"
    if vol_16 < vol_max:
        source_well_spot = "13"
    if vol_max<= vol_16 < vol_max*2:
        source_well_spot = "14"
    if vol_max*2 <= vol_16 < vol_max*3:
        source_well_spot = "15"
    if vol_max*3 <= vol_16 < vol_max*4:
        source_well_spot = "16"
    if vol_max*4 <= vol_16 < vol_max*5:
        source_well_spot = "17"
    if vol_max*5 <= vol_16 < vol_max*6:
        source_well_spot = "18"
    if vol_max*6 <= vol_16 < vol_max*7:
        source_well_spot = "19"
    if vol_max*7 <= vol_16 < vol_max*8:
        source_well_spot = "20"
    if vol_max*8 <= vol_16 < vol_max*9:
        source_well_spot = "21"
    if vol_max*9 <= vol_16 < vol_max*10:
        source_well_spot = "22"
    if vol_max*10 <= vol_16 < vol_max*11:
        source_well_spot = "23"
    if vol_16 >= vol_max*11:
        source_well_spot = "24"
if temp_well == 24:
    vol_24 += volume_list[23]
    vol = volume_list[23]
    source_well_section = "L"
    if vol_24 < vol_max:
        source_well_spot = "13"
    if vol_max <= vol_24 < vol_max*2:
        source_well_spot = "14"
    if vol_max*2 <= vol_24 < vol_max*3:
        source_well_spot = "15"
    if vol_max*3 <= vol_24 < vol_max*4:
        source_well_spot = "16"
    if vol_max*4 <= vol_24 < vol_max*5:
        source_well_spot = "17"
    if vol_max*5 <= vol_24 < vol_max*6:
        source_well_spot = "18"
    if vol_max*6 <= vol_24 < vol_max*7:
        source_well_spot = "19"
    if vol_max*7 <= vol_24 < vol_max*8:
        source_well_spot = "20"
    if vol_max*8 <= vol_24 < vol_max*9:
        source_well_spot = "21"
    if vol_max*9 <= vol_24 < vol_max*10:
        source_well_spot = "22"
    if vol_max*10 <= vol_24 < vol_max*11:
        source_well_spot = "23"
    if vol_24 >= vol_max*11:
        source_well_spot = "24"
```

```
                if temp_well == 32:
                    vol_32 += volume_list[31]
                    vol = volume_list[31]
                    source_well_section = "P"
                    if vol_32 < vol_max:
                        source_well_spot = "13"
                    if vol_max<= vol_32 < vol_max*2:
                        source_well_spot = "14"
                    if vol_max*2 <= vol_32 < vol_max*3:
                        source_well_spot = "15"
                    if vol_max*3 <= vol_32 < vol_max*4:
                        source_well_spot = "16"
                    if vol_max*4 <= vol_32 < vol_max*5:
                        source_well_spot = "17"
                    if vol_max*5 <= vol_32 < vol_max*6:
                        source_well_spot = "18"
                    if vol_max*6 <= vol_32 < vol_max*7:
                        source_well_spot = "19"
                    if vol_max*7 <= vol_32 < vol_max*8:
                        source_well_spot = "20"
                    if vol_max*8 <= vol_32 < vol_max*9:
                        source_well_spot = "21"
                    if vol_max*9 <= vol_32 < vol_max*10:
                        source_well_spot = "22"
                    if vol_max*10 <= vol_32 < vol_max*11:
                        source_well_spot = "23"
                    if vol_32 >= vol_max*11:
                        source_well_spot = "24"
                temp_char = "Master {0}{1} {2} {3}{4} {5}".format(source_well_section,
source_well_spot, dest_plate, dest_well_section, dest_well_spot, vol)
                char_list[char_spot+6] = temp_char
            char_var += 1
            char_counter += 1
        else:
            continue

    if char_var >= len(list): #piecing together all the rows into one large list
        for i in range(len(char_list)):
            if i >= len(char_list):
                break_var = True
                break
            if char_list[i] == "`":
                char_list.pop(i)
                if i >= len(char_list):
                    break
                while char_list[i] == "`":
                    char_list.pop(i)
                    if i >= len(char_list):
                        break
            if i >= len(char_list):
                break_var = True
                break
            else:
                continue

        vol_list.append(vol_1) #Adding all total volumes used into one list
```

```
        vol_list.append(vol_2)
        vol_list.append(vol_3)
        vol_list.append(vol_4)
        vol_list.append(vol_5)
        vol_list.append(vol_6)
        vol_list.append(vol_7)
        vol_list.append(vol_8)
        vol_list.append(vol_9)
        vol_list.append(vol_10)
        vol_list.append(vol_11)
        vol_list.append(vol_12)
        vol_list.append(vol_13)
        vol_list.append(vol_14)
        vol_list.append(vol_15)
        vol_list.append(vol_16)
        vol_list.append(vol_17)
        vol_list.append(vol_18)
        vol_list.append(vol_19)
        vol_list.append(vol_20)
        vol_list.append(vol_21)
        vol_list.append(vol_22)
        vol_list.append(vol_23)
        vol_list.append(vol_24)
        vol_list.append(vol_25)
        vol_list.append(vol_26)
        vol_list.append(vol_27)
        vol_list.append(vol_28)
        vol_list.append(vol_29)
        vol_list.append(vol_30)
        vol_list.append(vol_31)
        vol_list.append(vol_32)
        for i in range(len(vol_list)): #printing all total volumes at the end
            print(vol_list[i])
        for i in range(len(char_list)): #adding the list of rows to the input to Echo file
            ws.write('A{0}'.format(i+1),char_list[i])
        print("Done")

        wb.close()
        exit()


try:
    setup()
    while True:
        loop() #looping main section of code

except Exception as reason: #this has python print out the reason the program crashed if the code was interupted suddenly
    if len(reason.args)>0 and reason.args[0] == "User quit the game":
        print("Crash.")
    else:
        print(reason.args)

df.close()
df_2.close()
```

Computer Program 3 | "Molbit Decoding"

```
% ------------------------------------
% Groups molbits for assignment to individual bytes and generates a table for the "new profiler" program
% [This code was run in Python]

import csv, math
import pandas as pd
from openpyxl import Workbook
import xlsxwriter


def setup():
  global df, csv_df, var, list, break_var, char_list, byte, spot, char_var, i, df_2, csv_df_2, masses_list, mass_counter, temp_counter, ws, wb #makes variables global so they can be used in other methods
  global dest_well_section, dest_well_spot
  df = open("Ascii binary combinations 3.csv") #opens csv files in folder so the can be manipulated
  df_2 = open("Peptide Masses.csv")

  csv_df = csv.reader(df)
  csv_df_2 = csv.reader(df_2)

  wb = xlsxwriter.Workbook('decryption.xlsx')
  ws = wb.add_worksheet("New Sheet")

  i = 0 #defining all variables
  dest_well_spot = 0
  dest_well_section = 0
  mass_counter = 0
  temp_counter = 0
  char_list = []
  char_var = 0
  byte = 0
  spot = 0
  break_var = False
  var = input("Type Here -->") #asking for what is to be decrypted
  # print(var)
  list = [c for c in var] #splitting up input into a list of each character
  masses_list = []
  for row in csv_df_2:
    masses_list.append(row)
  # print(list)

  while i < len(list):
    char_list.append("0")
    i += 1
    # print(char_list)


def loop():
  global break_var, char_list, byte, spot, char_var, i, mass_counter, temp_counter, dest_well_section, dest_well_spot

  for row in csv_df: #looping through all characters in csv file
```

```
    for char in range(len(list)): #looping through all characters in the inputed list
        if row[0] == list[char]:

            dest_well_spot = "13" #calculating what the destination plate row is
            dest_well_section = "A"
            if 25 <= math.floor(char / 4) + 1 < 49:
                dest_well_section = "B"
            if 49 <= math.floor(char / 4) + 1 < 73:
                dest_well_section = "C"
            if 73 <= math.floor(char / 4) + 1 < 97:
                dest_well_section = "D"
            if 97 <= math.floor(char / 4) + 1 < 121:
                dest_well_section = "E"
            if 121 <= math.floor(char / 4) + 1 < 145:
                dest_well_section = "F"
            if 145 <= math.floor(char / 4) + 1 < 169:
                dest_well_section = "G"
            if 169 <= math.floor(char / 4) + 1 < 193:
                dest_well_section = "H"
            if 193 <= math.floor(char / 4) + 1 < 217:
                dest_well_section = "I"
            if 217 <= math.floor(char / 4) + 1 < 241:
                dest_well_section = "J"
            if 241 <= math.floor(char / 4) + 1 < 265:
                dest_well_section = "K"
            if 265 <= math.floor(char / 4) + 1 < 289:
                dest_well_section = "L"
            if 289 <= math.floor(char / 4) + 1 < 313:
                dest_well_section = "M"
            if 313 <= math.floor(char / 4) + 1 < 337:
                dest_well_section = "N"
            if 337 <= math.floor(char / 4) + 1 < 361:
                dest_well_section = "O"
            if 361 <= math.floor(char / 4) + 1 < 385:
                dest_well_section = "P"
                last_section = True

            if math.floor(char / 4) % 24 + 1 < 10: #calculating what the destinaition plate column will be
                dest_well_spot = "0{0}".format((math.floor(char / 4) % 24) + 1)
            else:
                dest_well_spot = (math.floor(char / 4) % 24) + 1
            temp_list = [c for c in row[1]]

            byte = (char % 4) #calculating byte number
            a_byte = byte + 1
            spot = math.floor(char / 4) + 1
            mass_list = []

            if temp_list[1] == "1": #getting what the appropriate mass values should be at that given
spectra
                mass_counter += 1
                mass_spot = 1 + (8*byte)
                mass_list.append(masses_list[mass_spot])
            if temp_list[2] == "1":
                mass_counter += 1
                mass_spot = 2 + (8*byte)
```

```
            mass_list.append(masses_list[mass_spot])
        if temp_list[3] == "1":
            mass_counter += 1
            mass_spot = 3 + (8*byte)
            mass_list.append(masses_list[mass_spot])
        if temp_list[4] == "1":
            mass_counter += 1
            mass_spot = 4 + (8*byte)
            mass_list.append(masses_list[mass_spot])
        if temp_list[5] == "1":
            mass_counter += 1
            mass_spot = 5 + (8*byte)
            mass_list.append(masses_list[mass_spot])
        if temp_list[6] == "1":
            mass_counter += 1
            mass_spot = 6 + (8*byte)
            mass_list.append(masses_list[mass_spot])
        if temp_list[7] == "1":
            mass_counter += 1
            mass_spot = 7 + (8*byte)
            mass_list.append(masses_list[mass_spot])
        temp_char = list[char]
        if mass_counter == 7: #splitting up the gathered information into the list that is needed for the
input
            temp = " {0} {3}{4} {1} {2}".format(temp_char, a_byte,
listToStringWithoutBrackets(mass_list[0]), dest_well_section, dest_well_spot)
            temp_2 = "{0} {1}".format(temp, listToStringWithoutBrackets(mass_list[1]))
            temp_3 = "{0} {1}".format(temp_2, listToStringWithoutBrackets(mass_list[2]))
            temp_4 = "{0} {1}".format(temp_3, listToStringWithoutBrackets(mass_list[3]))
            temp_5 = "{0} {1}".format(temp_4, listToStringWithoutBrackets(mass_list[4]))
            temp_6 = "{0} {1}".format(temp_5, listToStringWithoutBrackets(mass_list[5]))
            final_temp = "{0} {1}".format(temp_6, listToStringWithoutBrackets(mass_list[6]))
        if mass_counter == 6:
            temp = " {0} {3}{4} {1} {2}".format(temp_char, a_byte,
listToStringWithoutBrackets(mass_list[0]), dest_well_section, dest_well_spot)
            temp_2 = "{0} {1}".format(temp, listToStringWithoutBrackets(mass_list[1]))
            temp_3 = "{0} {1}".format(temp_2, listToStringWithoutBrackets(mass_list[2]))
            temp_4 = "{0} {1}".format(temp_3, listToStringWithoutBrackets(mass_list[3]))
            temp_5 = "{0} {1}".format(temp_4, listToStringWithoutBrackets(mass_list[4]))
            final_temp = "{0} {1}".format(temp_5, listToStringWithoutBrackets(mass_list[5]),
dest_well_section, dest_well_spot)
        if mass_counter == 5:
            temp = " {0} {3}{4} {1} {2}".format(temp_char, a_byte,
listToStringWithoutBrackets(mass_list[0]), dest_well_section, dest_well_spot)
            temp_2 = "{0} {1}".format(temp, listToStringWithoutBrackets(mass_list[1]))
            temp_3 = "{0} {1}".format(temp_2, listToStringWithoutBrackets(mass_list[2]))
            temp_4 = "{0} {1}".format(temp_3, listToStringWithoutBrackets(mass_list[3]))
            final_temp = "{0} {1}".format(temp_4, listToStringWithoutBrackets(mass_list[4]))
        if mass_counter == 4:
            temp = " {0} {3}{4} {1} {2}".format(temp_char, a_byte,
listToStringWithoutBrackets(mass_list[0]), dest_well_section, dest_well_spot)
            temp_2 = "{0} {1}".format(temp, listToStringWithoutBrackets(mass_list[1]))
            temp_3 = "{0} {1}".format(temp_2, listToStringWithoutBrackets(mass_list[2]))
            final_temp = "{0} {1}".format(temp_3, listToStringWithoutBrackets(mass_list[3]))
        if mass_counter == 3:
```

```
                temp = " {0} {3}{4} {1} {2}".format(temp_char, a_byte,
listToStringWithoutBrackets(mass_list[0]), dest_well_section, dest_well_spot)
                temp_2 = "{0} {1}".format(temp, listToStringWithoutBrackets(mass_list[1]))
                final_temp = "{0} {1}".format(temp_2, listToStringWithoutBrackets(mass_list[2]))
            if mass_counter == 2:
                temp = " {0} {3}{4} {1} {2}".format(temp_char, a_byte,
listToStringWithoutBrackets(mass_list[0]), dest_well_section, dest_well_spot)
                final_temp = "{0} {1}".format(temp, listToStringWithoutBrackets(mass_list[1]))
            if mass_counter == 1:
                final_temp = "{0} {3}{4} {1} {2}".format(temp_char, a_byte,
listToStringWithoutBrackets(mass_list[0]), dest_well_section, dest_well_spot)
            char_list[char] = final_temp
            # print(row[0], " ", row[1])
            # print(char_list)
            char_var += 1
            mass_list.clear()
            mass_counter = 0
            temp_counter = 0
        else:
            continue

  if char_var >= len(list): #writing the lists made into the excel sheet
      for i in range(len(char_list)):
          ws.write('A{0}'.format(i+1),char_list[i])
      wb.close()
      exit()

def listToStringWithoutBrackets(list1): #taking out brackets so the formating is correct
  return str(list1).replace("['",'').replace("']",'')


try:
  setup()
  while True:
      loop()

except Exception as reason:
  if len(reason.args)>0 and reason.args[0] == "User quit the game":
      print ("Crash.")

df.close()
df_2.close()
```

Computer Program 4 | "Image Encoding"

```
% ------------------------------------
% Encode image as a bitstream
% [This code was run in Matlab R2015b]

% optional: read the image and display it on screen
img = imread('filename.jpg');
imshow(img);

% open the original image file, convert to vector of bytes
fid = fopen('filename.jpg', 'r');
content = fread(fid, [1 inf], '*uint8');
fclose(fid);

% manipulate the file, which is initially in a vector of bytes (0-255)
% convert to the equivalent binary representation
b = de2bi(content);
c = fliplr(b);
d = c';

% this is the vector of binary digits (bitstream) to be encoded
result = d(:)';

% optional: convert to string
string = mat2str(result,1);

% determine lengthy of the bitstream, in bits
length = length(result);
```

Computer Program 5 | “Image Reconstitution”

```
% -------------------------------------
% Reconstitute image from a bitstream
% [This code was run in Matlab R2015b]

% open the experiment text file, convert to vector of bytes
% note that this file is an uninterrupted bitstream
fid = fopen('experimental_data.txt', 'r');
content = fgetl(fid);
fclose(fid);

% format text as a vector of bits
char_cells = num2cell(content);
all_data = cellfun(@str2num, char_cells(:,1:end));

% extract file length data
length_bin = all_data(1:16);
length_dec = bi2de(fliplr(length_bin));

% extract experimental data of correct length
exp_data = all_data(17:17+length_dec-1);
exp_back1 = vec2mat(exp_data,8);
exp_back2 = fliplr(exp_back1);
exp_back3 = uint8(exp_back2); %cast as uint8
content_back = bi2de(exp_back3)';

% finally, re-write the data back into an image:
fid = fopen('filename _reconstituted.jpg', 'w');
fwrite(fid, content_back);
fclose(fid);
```