

## Physics 209: Assignment #8

The point of this Assignment is to give you some practice entering a moderately elaborate program into the HP20S and also to give you a feeling for what can happen when you repeat a simple process over and over. If you are using a computer or some calculator other than the HP20S then you will have to figure out for yourself from the program listing and comments how to produce an analogous program on your own machine. You can write a much simpler program than mine if you are doing it with a computer but doing it on a programmable calculator can be rather tricky.<sup>1</sup>

The program is designed to help you solve the following puzzle:

Take a list of ten one-digit numbers. For example, 2503856200. In this list the digit 0 appears 3 times, 1 appears 0 times, 2 appears 2 times, 3 appears 1 time, 4 appears 0 times, 5 appears 2 times, 6 appears 1 time, 7 appears 0 times, 8 appears 1 time, and 9 appears 0 times. We can summarize this information in a new list: 3021021010, where the first number on the new list gives the number of 0's in the old list, the second lists the number of 1's, the third the number of 2's, etc. The new list is said to describe the old list.

The puzzle is to find a list of ten one-digit numbers that describes itself; i.e. the first digit on the list gives the total number of 0's appearing on the list itself, the second digit gives the number of 1's appearing on the list itself, the third the number of 2's, etc. Another way to put it is that if you have a self-describing list, then if you go through the process we went through to get 3021021010 from 2503856200 but you start with the self-describing list, then you will get back the very same list that you started with.

A closely related puzzle is to find a pair of lists, each of which describes the other. If you take the first list and go through the process I've described you will produce the second list. If you next follow that procedure with the second list you will get back the first. So each list describes the other.

There is an amazingly simple way to find such lists. By "amazingly simple" I mean this: I cannot remember the solution to either puzzle, and finding them by logical analysis is rather hard work, but I do remember a very simple procedure that will find a solution with no mental effort at all! This procedure can be carried out with pencil and paper with a certain amount of labor. But it can be done with comparative ease with the help of the HP20S.

---

<sup>1</sup> If you would prefer to use a programmable calculator that you already own, the program listing on page 8 ought to give you some indication of how much trouble you can expect translating my programs into the language of your own calculator. My guess is that it will be more than \$35 worth of trouble, but it's up to you.

In the simple procedure you start with any list of ten digits at all.<sup>2</sup> Call the list  $x_1$ . Then write down the list that describes  $x_1$ . Call it  $x_2$ . Next write down the list,  $x_3$ , that describes  $x_2$ , and just keep on going.

The technical term for keeping on going is “iterating”. What I have just described is known as an “iterative procedure.” Each step in the process (going from one list to the next one) is called an “iteration”. Since there is only a finite (though huge<sup>3</sup>) number of possibilities for a set of ten digits, eventually you will get back to a list that you have already encountered, after which you just cycle through the old pattern that starts at that list over and over. If there are just  $n$  lists in that pattern, then the group of lists making up the pattern is called an  $n$ -cycle. You can arrange such lists on a circle, and as you go around the circle each list describes the one before it. A self describing list is just a 1-cycle, sometimes also called a “fixed point”.

*Question 1.* Try it yourself, working out  $x_2$ ,  $x_3$ , etc. for the example in the preceding paragraph. Don’t hand in your working paper; just give the list  $x_1, x_2, x_3, x_4, \dots$  that you found and describe in a few sentences the behavior you discovered.

As you probably noticed in answering Question 1, the process of calculating  $x_{i+1}$  from  $x_i$  (“iterating”) can get a bit tedious, and it is easy to make mistakes. A computer can make this much easier. On the last page I list a 35 step program that does the job for you. The program represents each list of ten digits by a decimal number between 0 and 1, written out to ten decimal places (including, if necessary, enough zeros at the end to be sure the number has exactly ten digits). For example the two lists described above would be represented in the calculator by 0.2503856200 and 0.3021021010. There are three minor technical points to note:

First, the 0 to the *left* of the decimal point is of no interest and is not considered to be part of the list. (I know of no way to turn it off; just ignore it.)

Second, you have to use the calculator in a mode in which it displays ten decimal digits to the right of the decimal point, rather than rounding the number off to a smaller number of digits. How to make the calculator do this is described in paragraph 7 on page 5.

Third, another idiosyncrasy of the calculator is that it does not display final zeros

---

<sup>2</sup> Well, not quite *any* list. If all ten digits are the same then that particular digit occurs 10 times. Since 10 is not a single digit, that particular list cannot be described by a second list of single digits and the process breaks down. And if all ten digits are different then after the first step you get a list of ten 1’s, after which the process again breaks down. I believe these are the only exceptions.

<sup>3</sup> Actually if you think about it you can cut down the number of possibilities. For example after the first and all subsequent iterations, the sum of all the digits on the list must be 10.

except when you first enter a decimal number.<sup>4</sup> Thus it prefers to display .2503856200 as .25038562, without the final two 0's. Therefore in writing down the result the calculator shows you after each iteration, you have to add enough zeros on the right to bring the number up to ten digits. Note that the calculator does count the zeros—it just doesn't display them.

*Question 2.* Enter into your HP20S the program<sup>5</sup> listed on the last page. When you run it it replaces whatever list is in the display with a new list that describes the old list. Experiment with a variety of starting lists<sup>6</sup>  $x_1$ , and describe what you discover when you run the program repeatedly. Can you find a fixed point? Can you find more than one fixed point? Can you find a cycle of length longer than 1? For a given fixed point or cycle, typically how long does it take before you settle down (the technical term is “converge”) to the fixed point or cycle. Of the various lists you started with, which takes the greatest number of steps before you converge to a fixed point or cycle?

*Question 3. Some open ended explorations.* I've been able to find a fixed point and a 2-cycle. As I remember, the 2-cycle is easier to find than the fixed point. I've never been able to find anything else, but maybe I've been unlucky in my guesses. One thing you can investigate is the “stability” of the fixed point and the 2-cycle. By this I mean the following: Start with a list that is close to the fixed point (or 2-cycle) in that it differs only in a single digit. When you iterate from such a starting point do you get back to the fixed point (or 2-cycle) or do you go over to the 2-cycle (or fixed point or something else)? Try this for several different examples. Can you find examples of both types of behavior (changing a fixed point to a 2-cycle or changing a 2-cycle to a fixed point) depending on which digit you alter and what you change it into?<sup>7</sup> More generally (this is an open-ended question) you might want to investigate the “basins of attraction” of the fixed point and 2-cycle. A list is in the basin of attraction of a fixed point or two cycle if, starting with that list, you get to the fixed point or two cycle. Which seems to have the larger basin of attraction? Any other interesting discoveries?

---

<sup>4</sup> Presumably the reason it allows you the final zeros when you enter a number is that it has no way of knowing until you've finished whether you intended to put a non-zero digit in the last position.

<sup>5</sup> See below for how to do this. These instructions will also be useful for entering the other programs we will be using, so save this assignment.

<sup>6</sup> Remember that a list must begin with a decimal point if the program is to do its job. To automate the effort you expended on Question 1 you would run the program after entering .2503856200 in the display.

<sup>7</sup> I haven't tried this myself, so I don't know the answer.

## How to get your HP20S to run the program on page 8.

A summary of the information needed to enter, run, and debug a program can be found in the HP20S Owner's Manual on pages 67–72. My own recipe book for what you should do is as follows:<sup>8</sup>

1. Note that most buttons have white labels on them and blue and yellow labels above them. Each button can function in three different ways. To get it to do the yellow or blue thing you press the button with the yellow (right pointing) or blue (left pointing) arrow before pressing the button itself. If you just press the button without pressing one of the arrow buttons first, it does the white thing. (The keys in the top row have a fourth function: the little letter (A–F) below them and to the right serve as program labels, as we shall see.)

2. The button with the big C at the lower left turns on the calculator. Since OFF is written above it in yellow, to turn the calculator off you press the yellow button and then press button C. If you press C when the calculator is on it simply replaces the displayed number with 0. (If you ever get an error message you can also get things started again by pressing C.)

3. Keys are conveniently described by giving the number of the row they are in, followed by their position in that row. Thus the key C is described by 71. The blue and yellow arrows are 51 and 61. The key 8 is key 43, etc.<sup>9</sup> Since the OFF operation is achieved by pressing first the yellow key and then key C, OFF is described by the symbol 61 71.<sup>10</sup>

4. To enter a program, switch the calculator to programming mode with the key PRGM, which is the R/S key in its blue mode, described as 51 26. (Note that this description is useful in hunting down the keys mentioned.) If there is no program already in program memory you will see 00– in the display, which means the calculator is ready to start writing down the program as you enter it. If there is a program already there you can get rid of it (if you wish to) with the clear-program key, CLPRGM, which is the INPUT key operating in its yellow mode; i.e. first you press the yellow key and then the input key; in numbers this would be described as 61 31. WARNING: be careful not to clear a program accidentally after you have entered it! Fortunately you have to press two keys to do this, so it is not likely to happen unintentionally. If you want to keep the other program you can write your program below it, provided the two together don't take more than 100

---

<sup>8</sup> It is my hope that my recipe is clearer than the manual, which you need not consult at all unless you disagree.

<sup>9</sup> It is more convenient to refer to the number keys 0,1, . . . 9 by the numbers themselves rather than by their rows and positions.

<sup>10</sup> To save the battery the calculator automatically turns itself off if you don't do anything for several minutes. Don't panic. No programs are lost when this happens. Just turn it on again. If you were in programming mode (see 5 below) when it went off you will have to enter programming mode again when you turn it on, but it will then put you at the same point in the program where you were when it went off.

steps. To get to 00– all you do is press GTO (the XEQ key in its blue mode— 51 41 or, equivalently, blue 41) followed by two decimal points: GTO .. (51 41 73 73).<sup>11</sup> As you write the new program starting at 00, the steps of the old program will be automatically pushed higher in program memory. If they end up going higher than 99– they will be lost.<sup>12</sup>

5. Now start to enter the program as given on the last page. The first instruction is LBL A which you enter by pressing the LBL key (yellow XEQ) followed by key A in the top row. (When you press a top row key after pressing LBL the calculator interprets that top row key in its LBL mode; the same holds for any other key that has to be followed by a program label.) The program listing tells you explicitly that LBL A is 61 41 A—i.e. to enter LBL A you press those three keys. If you’ve done it correctly, the window will then show 01– 61 41 A. If it doesn’t, you made a mistake. In that case press the white arrow key (35) which deletes the program step enabling you to try again. If you were successful, however, you can now enter the second instruction: STO 0 (21 0). After you have done this the window will show 02– 21 0; if you made a mistake press key 35, which enables you to try again. Continue in this way until you have entered all 35 steps of the program.<sup>13</sup>

6. To check that you have entered the program correctly you can step through it. You do this by using the down arrow key (blue 7) (to go down the list of program steps) or the up arrow key (blue 8) (to go up the list). If you want you can just hold the blue key down steadily while stepping around with the 7 and 8 keys. It is crucial, however, not to forget to use the blue key or you will enter a series of nonsense program steps consisting of the numbers 7 and 8. (If you discover you have done this, however, they are easily deleted by means of the white arrow key (35) as described in step 5 above.) To step through the

---

<sup>11</sup> This works whether or not you are in programming mode. If you are not in programming mode and do GTO .. then when you switch to programming mode you will always see 00– in the display whether or not there are other things in the program memory.

<sup>12</sup> Alternatively you can write the new program *above* an already existing program. To do this go to program step 00– as described above, and then press the number key 8 in the blue mode (51 8). This takes you one step up, as indicated by the blue arrowhead pointing up above the 8-key. (Note that “One step” up in the program listing means the next lower numbered step, since the numbers increase as you go down the list. If I were Hewlett Packard I wouldn’t have adopted so potentially confusing a convention.) But when you ask the calculator to take you one step up from 0 (the highest step in the listing) it goes all the way to the bottom of the listing and therefore puts you at the highest numbered step — precisely where you want to be to add the instructions for the new program, if you want it to follow the last instruction in the old program.

<sup>13</sup> Don’t worry, unless you are genuinely curious, about what the program steps mean and how they manage to add up to something that does just what we want. I put the comments in to remind me what the program does, in case I want to revise or improve it, and because they might help you if you are trying to design a comparable program for a different calculator or a computer.

program you can either do GTO .. (51 41 73 73) which places you at position 00 (or, if you are at position 35, can do the step down key (blue 7) which, if there are no more program steps, cycles you back to 00. It's not a bad idea after entering the program to step through it to make sure you've entered everything correctly.

7. To run the program we can test it on the numerical example given on page 1. First you must leave program mode. You do this by pressing PRGM (blue R/S) (51 26) just as you did to enter it. To see what we are getting, we must use the calculator in a mode in which it displays all the digits of a number, rather than rounding them off to some smaller number of digits. We insure this by pressing the ALL key before we begin. After that the calculator will stay in the ALL mode until you tell it otherwise.<sup>14</sup> ALL is yellow right parenthesis (or 61 34). Now clear the display, if it does not already have a 0 in it, by pressing C. Next, enter the decimal number .2503856200. (Be sure to start with a decimal point.) You can start the program in either of two ways: (1) If you are sure the program counter is at position 0 just press the button R/S;<sup>15</sup> (2) You can simply do XEQ A, since the first instruction labeled the program A. If you entered the program correctly .302102101 will appear in the window after a little while. Note that the calculator does not list the final zero, so in writing down the result at each step you have to add enough zeros on the right to bring the decimal number up to ten digits. (The calculator does, however, count the zeros—it just doesn't like to display them.) If .302102101 did not appear, then you did not enter the program correctly. Do GTO .. (51 41 73 73), enter program mode, and step through the program checking it against the listing on the last page.<sup>16</sup> When you find the incorrect instruction(s) delete them with the white arrow key (35) and enter them correctly.

8. You are now in business, and can get to work on questions 2 and 3. Enter into the display a ten digit list starting with a decimal point, and run the program (XEQ A).<sup>17</sup> It will put into the display the list that describes the one you entered. To get the list that describes the new list in the display, simply press R/S. Continue pressing R/S until the lists settle down to a fixed point or a cycle. To start with a new list simply delete the list in the display with C (71), put in a new ten digit list starting with a decimal point, and

---

<sup>14</sup> ALL mode can be irritating for other purposes. The mode I recommend for purposes other than this particular program is the one that displays 9 decimal places. You get into it with blue FIX 9 (51 33 9).

<sup>15</sup> If you are not sure you can set it there by doing GTO .. (51 41 73 73).

<sup>16</sup> Make sure you keep that blue button down while pressing the down arrow key, or you will pollute your program with nonsense steps containing the number 7 (or 8, if you forget to press the blue button while pressing the up arrow key.) I already warned you about this but I remind you again, because it's very irritating when it happens.

<sup>17</sup> You only have to do XEQ A when running the program for the first time. After that the single button R/S is enough, unless you've run some other program since last running this one, or have otherwise managed to go to some other location in program memory.

press R/S. Take notes on what you discover.

9. Remarks about the program listing: there are 4 columns. The first is just the number of the program step (which the calculator will display in the form 01-, 02-, etc.) The second is the key (or keys) you have to press to enter the program step. (I've preceded those steps that require a key to be in blue or yellow mode, with  $[b]$  or  $[y]$ .) The third column tells you the sequence of keys in terms of the numerical code that locates them. This is useful (a) to help you find where the key is on the calculator and (b) because this is what appears on the screen if you have correctly entered the step, which makes it easy to check for errors. Finally, the 4th column comments on what the step does. You don't have to worry about this unless you are curious about how the program actually works or you want to write an analogous program on a programmable calculator or computer other than the HP20S. If you do use a computer it ought to be possible to write a program considerably simpler than mine.

## Listing for the Puzzle Program on the HP20S

No.	Key name	Key code	Comments
1	[y] <b>LBL A</b>	61 41 A	Labels program A
2	<b>STO 0</b>	21 0	Stores current list in location 0
3	.	73	Stores number 0.1
4	<b>1</b>	1	in location 2
5	<b>STO 2</b>	21 2	for future use
6	<b>1/x</b>	15	Converts 0.1 to 10
7	<b>STO 3</b>	21 3	Stores 10 in location 3 for use in shifting left
8	<b>STO 4</b>	21 4	Stores 10 in location 4 for use as counter
9	<b>0</b>	0	Stores 0 in location 1 where we
10	<b>STO 1</b>	21 1	will build up the next list
11	<b>XEQ B</b>	41 B	Transfers to subroutine B to compute next list
12	<b>RCL 1</b>	22 1	Puts next list in display
13	<b>R/S</b>	26	Stops so list can be viewed
14	[b] <b>GOTO A</b>	51 41 A	Starts over again at A to calculate next list when R/S is pressed
15	[y] <b>LBL B</b>	61 41 B	What follows is labeled program B
16	<b>RCL 0</b>	22 0	Copies current list from location 0
17	<b>×</b>	55	Multiplies it by 10 (which shifts)
18	<b>RCL 3</b>	22 3	everything one place to left
19	<b>=</b>	74	and puts result
20	<b>STO 0</b>	21 0	back in location 0
21	[b] <b>IP</b>	51 45	Takes first digit on left
22	<b>STO - 0</b>	21 65 0	and removes it from number in location 0
23	$y^x$	14	Raises 0.1 (taken from location 2) to power
24	<b>RCL 2</b>	22 2	of this digit and then shifts
25	[b] <b>SWAP</b>	51 31	result one more place to right
26	<b>×</b>	55	which has effect of putting a 1
27	<b>RCL 2</b>	22 2	in position on list corresponding
28	<b>=</b>	74	to that digit
29	<b>STO + 1</b>	21 75 1	Adds that 1 to appropriate place on new list
30	<b>1</b>	1	Reduces counter in 4 (originally 10)
31	<b>STO - 4</b>	21 65 4	by 1 to indicate another step has been taken
32	<b>RCL 4</b>	22 4	Looks to see how many steps are left
33	[y] $x = 0?$	61 43	Asks: are we finished?
34	[y] <b>RTN</b>	61 26	If so return to where we left main program
35	[b] <b>GOTO B</b>	51 41 B	If not, go do this subroutine again