For each of the algorithms listed below, create pseudocode for the algorithm and a flow chart for the algorithm. You may choose a problem to 'tailor' your pseudo-code to. The Travelling Salesman Problem is always an option. I mentioned in class, that you should have functions for the various parts of the metaheuristic, such as initialize, mutate, change_temperature and so forth. Each function should be tailored for the problem you choose.

Please submit your work in a Microsoft Word or Google Doc format.

1. Simulated Annealing
2. Genetic Algorithms

For the flowcharts: article [1] has some pointers, if you need. [2] is a link to free flowcharting software. [3] is a link to my video on how to create flowcharts using Excel.

For the pseudocode: The definition we will use is "a notation resembling a simplified programming language, used in program design" [Google]. I want you to use a 'Python style' pseudocode. [4] gives an explanation of pseudocode as well as some examples of different styles of pseudocode. Since you know Python, we will use that as a basis for the pseudocode. Please see below for pointers and guidelines for the pseudocode.

[1] https://www.programiz.com/article/flowchart-programming

[2] https://www.lucidchart.com/pages/landing/flowchart-program

[3] https://youtu.be/Phgk6XX9hRw

[4] https://en.wikipedia.org/wiki/Pseudocode

Pseudocode is a way to document, and explain algorithms. With pseudocode, we should be able to read the code and understand (hopefully easily) how the algorithm works. A primary characteristic of good pseudocode is clarity. The following is to help you create good pseudocode. You may comment your pseudocode, but you shouldn't have to. While we are using syntax similar to Python, it is certainly acceptable, and perhaps preferable, to use a sentence rather than code especially if the sentence is clear.

Declare variables

Indicate what the variable is for and what type it is. It is best to use descriptive variable names to make the pseudocode more easily understandable, which is the point. Camel Case is encouraged (https://en.wikipedia.org/wiki/Camel_case). FirstName

Examples:

Declare integer index # index is the loop counter

Declare integer upperLimit # upperLimit is the limit of the loop

Declare a real constant pi # pi is the value of pi

Declare a string variable called userName # userName holds the username

Assignment statements

would look like this:

index = index + 2

If-then-else

For flow control Python uses if, elif, else and : (colon). We will substitute endif for the colon for clarity.

```
if index == 0:
return index
elif index > 1 and index mod 2 == 0:
return 'even'
else
        Do something else
endif
```

Loops

We will use the while loop as an example.

Python uses while and : (colon). We will add the end while to make program flow clear.

```
while index < upperLimit:
Name = WindowsCall(usrname)
if index == 0:
return index
```

```
elif index > 1 and index mod 2 == 0:
return 'even'
else
                    Do something else
Endif
index = index + 1
end while (or wend)
```

Arrays/Lists

A Python list L starts at index 0 and goes to index len(L) − 1. Python also allows for a

negative index as in L[−j], where 1 _ j _ len(L), which refers to the j-th element from the end of list L. We will use the Python slice notation, where L[i : j] means the part of the list L from index i to index j − 1, including L[i] but not L[j].

E.g., let P be an array/list containing numbers [2, 3, 5, 7, 11, 13, 17, 19].

Strings

We will think of Python strings as lists of characters.

S[j] = 'C' is fine.

Multiple statements in a line

For program clarity, don't.

Indenting

Blocks of code must be indented. The examples above should serve.