CSC180F: Final Exam: 2017-12-13

First name (please write as legibly as possible within the boxes)

Last name

Student ID

Instructions

- You MUST use pen. Pencil is NOT permitted on this exam.
- Write your official first name, last name, student number and utorid, using ONLY bold capital (UPPERCASE) letters and numbers.
- Closed book; no aids; no electronic equipment allowed (cellphones, tablets, computers, calculators, etc.).
- Attempt and answer ALL questions for a total of **160 points**
- If a particular question seems unclear, explicitly state any reasonable assumptions and proceed with the problem.
- Show all steps and present solutions clearly

Possibly useful information about Python

```
Types: sequential (list, string), numeric (integer, float),
       boolean (True, False)
Operators: tests (==, <>, <=, >=, >, <), and, or, not,
           +, -, *, /, %, assignment (=), list copy (L=list(M))
Extending the language: functions (def ... return), classes
```

Possibly useful information about C

```
Types: numeric (int, float), char, struct,
       arrays (TYPE NAME[SIZE]), pointer/address (TYPE *NAME),
Operators: tests (==, !=, <=, >=, >, <), &&, ||, !, +, -, *, /, %,
           assignment (=)
Extending the language: functions structs

Type Analysis:
  TYPE *X; means: *X is a TYPE
                   X is a TYPE *
                  &X is a TYPE **

Passing pointers:
 if the function is defined: RVAL_TYPE FNAME(TYPE1 *PARG1_NAME,...);
 if the caller defines:
    TYPE1 X;
    RVAL_TYPE R;
 then the caller must call FNAME via:
    R = FNAME( &X, ...)

 Since PARG1_NAME is a TYPE1 *, then inside of FNAME you can write:
     *PARG1_NAME = <some value>;
 and the value of X in the caller (which is outside of FNAME) will be set
 to that value.

 If you want to return a value of type TYPE via the arg list,
 then you must define the argument for that value as a TYPE *

Malloc syntax: if you define TYPE *X;
 then you can allocate memory for X via:
   X = (TYPE *) malloc(sizeof(TYPE) * <SIZE>);

 Then X[0] refers to a value of type TYPE at the start of that memory.
      X[<SIZE>-1] refers to the final value of type TYPE in that memory
```

1. **(Pts as shown)** A number is a BLAH number if it is divisible by any of the odd numbers below it (e.g., 9 is a BLAH number because it is divisible by 3; 8 is not a BLAH number because none of the odd numbers below 8 will divide 8; 7 is not a BLAH number because it is not divisible by any number below it).

   a) **(10 pts)** Write a function (isBlah) in Python that returns True if its integer argument is a BLAH number; otherwise it should return False.

b) **(10 pts)** Write a function in C (isBlah) that returns 0 if its integer argument is a BLAH number; otherwise it should return -1.

c) **(10 pts)** Write a function in Python that will take an argument n and then return a list of the first n BLAH numbers (starting from, but not including 1).

d) **(10 pts)** Write a function in C that will take an argument n and then return an array of the first n BLAH numbers (starting from, but not including 1).

e) **(10 pts)** Write a function in C that will take in two arguments (n, output); n is the number of blah numbers to return, and output is a caller-allocated argument variable that will be used to transfer n blah numbers (starting from, but not including 1) back to the caller. The return value of the function will be 0 or -1; 0 is returned if there are no errors, and -1 is returned if there is an error.

f) **(15 pts)** Write a function in C that will take in two arguments (n, output); n is the number of blah numbers to return, and output is an argument variable (not allocated by the caller) that will be used to transfer n blah numbers (starting from, but not including 1) back to the caller. The return value of the function will be 0 or -1; 0 is returned if there are no errors, and -1 is returned if there is an error.

2. **(Pts as shown)** You are required to write a program to perform symbolic calculus on single variable polynomials. These polynomials will consist of multiple terms; you do not know in advance how many terms are present in the polynomial. You may assume the exponents are non-negative.

   a) **(5 pts)** Show how you would define, in C, the data structure to represent such a multi-term, single variable polynomial.

b) **(15 pts)** Write a C function (integrate) that takes a single argument which is a pointer to the data structure defined above. It will integrate the input polynomial and modify this input data structure so that it will contain the symbolic integral (i.e., there is no separate data structure to represent the output; you will re-use the input data structure for this purpose). You may ignore the fact that integration always produces a final added constant. The return value of the function is 0 for success, and -1 for error.

3. **(10 pts)** Write a Python function (srs) that will take a single argument n, and return the sum of the reciprocals of the first n squares. E.g., srs(3) means to return the sum $\frac{1}{1} + \frac{1}{4} + \frac{1}{9}$ which is 1.361111...

4. **(10 pts)** Write a Python function (dig2str) that will take a single argument n which is a digital from 0 to 5 (inclusive), and return a string corresponding to the word that describes the digit. E.g., dig2str(4) would return "four".

5. **(15 pts)** Write a Python function (repeater) that will take a single argument n which is a string, and returns a new string that consists of the letters of the original string, in which the letter at position i is repeated $i + 2$ times, in order. E.g., repeater("abc") would return "aabbbcccc".

6. **(15 pts)** Write a Python class (rotater) whose init function will take in a single additional argument which is a string; an internal variable (store) will be set to that string. There is also a function in this class called evolve which will take in a single additional argument n that specifies how many rotations should be applied to store. E.g., if store was "abcd" and evolve(1) was called, then store would be updated to "bcda". If evolve(1) were the called, then store would be "cdab" (from "bcda"). If evolve(2) were then called, then store would be "abcd" (from "cdab"). Evolve should both update store, and return the updated value of store.

7. **(5 pts)** Show how you would define a data structure in C to handle four-variable multi-term polynomials.

8. **(20 pts)** Consider a function in C called nsum that has four arguments: a (an input vector), b (another input vector), n and output. It should compute an output vector that contains the vector sum of a and b. To handle general numbers, what should the types of a and b be (3 points)? What is the purpose of n, and what is its type (2 points)? Assuming the caller did not allocate output, what is the type of output (3 points)? Write this function (12 points).