ESC180: TestB 2018-11-22 (90 minutes)

Instructions

- Don't detach the pages. Pen is recommended. If you use pencil, write very hard as the test will be scanned. Don't freak out, this is just a test.
- Write your official first name, last name, student number using UPPER-CASE letters and numbers.
- Closed book; no aids; no additional paper; no electronic equipment allowed.
- Attempt and answer ALL questions; there are 16 pages on this exam including three blank sheets.
- No questions will be answered by the examiner: if a question seems unclear, you may try to state assumptions that will allow you to proceed.
- Show all steps and present solutions clearly using the most appropriate C concepts from the lectures; non-course content will not be accepted.

1. **5 points** Write a function that will return the average of an array of floats; you must use the most general argument type for the array, and ensure that your code is safe (in the context of our lectures).

2. **5 points** Let `sizeof(int)=4`, and `sizeof(char)=1`; the computer we program has 8 byte addresses. What would `sizeof(struct blah)` be if we define:
`struct blah { int id; int val[4]; char label[4]; int *data; };`?

3. **5 points** For the same computer as the prior question, how many bytes are needed to store (a) a linked list of 100 integers, and (b) an array of 100 integers. Show all work.

4. **5 points** Let f be defined:

```
int f(int x) {
    if (x==-1) {
        return 2;
    } else {
        return x*f(x-2);
    }
}
```

   What is f(5)? Show all work.

5. **5 points** Consider two functions, `f` and `g`; `f` will declare a variable for an array of char elements, where the number of elements will never change, but it is not known a priori, and `g` will allocate the memory for the variable mentioned above (the amount of memory will be allocated based on an argument provided to g). Write the function declaration for `g`, and write the line of code in `g` that will do the allocation; `g`'s return value is 0 or -1 based on whether there is success or failure.

6. **(5 pts)** Show how you would define a data structure in C to handle four-variable multi-term polynomials.

7. **5 points** Consider the following program:

```
#include <stdio.h>
int main(void) {
    int *b;
    int **a;
    int c = 10;
    int d = 20;
    int q = 30;
    b=&c;
    a=&b;
    q=**a;
    *b=0;
    d = c + 100;
    q = q + 100;
    /* HERE */
    return 0;
}
```

Report, in alphabetical order, the values of all variables at the point `/* HERE */`; if something is an address, then say "points to VAR" where VAR is reported suitably.

8. 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.

9. **(15 points)** Consider the function whose prototype is

```
int fil(typeA matrixIn,
        typeB rows, typeC cols,
        typeD matrixOut);
```

whose action is to filter matrixIn by replacing all negative entries by 0, and output the result via matrixOut; returning 0 for success and -1 for failure. The elements of matrixIn will be floating point numbers.

a) (3 points) Assuming the use of C types that you learned in class (and not user-defined data structures via struct), what are typeA, typeB, and typeC?

b) (2 points) Assume that the caller of this function will provide an allocated data structure for matrixOut. What is typeD?

c) (2 points) Assume that the caller of this function will not provide an allocated data structure for matrixOut. What is typeD?

d) (8 points) Write this function in C, assuming the caller will not provide an allocated data structure for matrixOut. Include error checking.

10. **(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).

11. **(25 points)** Write a function, `isPrime(n)` that returns 0 if n is a Prime and -1 otherwise. Now write a second function `isPrimeProduct(n)` that returns 0 if n is a product of two prime numbers (-1 otherwise); if needed, use `isPrime` as a helper. Clarity and correctness are vital.

12. **(25 points)** Write a function, `histogram(n,m,s)`, where n and m are caller-allocated arrays of equal size, s, and n contains the input integer data for this problem. Array m is an array of structs, where each struct consists of three fields: valid (int), value (int), frequency (int). The frequency field reports how many times value was found in n. Report the frequencies in the order that the values appear in n. You may assume that the valid component for each element of m is set to 0 by the caller of histogram, initially.

13. **(5 points)** Write the function `getmode` that takes in two arguments: `m` and `mode`, where `m` is exactly as defined in the prior question, and `mode` is the output that `getmode` will set to the MODE (most frequent value) of the data set in `m`. Return values are either 0 (success) or -1 (failure).

14. **(10 points)** Write a function, `fibo(n)`, in C that, via recursion, returns the fibo3 sequence element at position n, where fibo3 is defined as per: fibo3(0)=1, fibo3(1)=1, fibo3(2)=1, fibo3(3)=3, fibo3(4)=5, fibo3(5)=9, ...

(Rough unmarked work)

(Rough unmarked work)

(Rough unmarked work)