CSE 333 – Autumn 2016
Midterm

Name: _____

Please do not read beyond this cover page until told to start.

Nothing in this midterm is about C++.

You don't have to explain your answer unless the question asks you to. If you're unsure of your answer, a little explanation could get partial credit if the answer is wrong. (A correct answer with an incorrect explanation will not get full credit.) Long explanations are a mistake, as they will take too much of your time and provide too much opportunity to get something wrong.

We will be using an online system to grade the midterms. That requires that we scan all the midterms. ***Please do not write on the backs of the pages, nor very close to the edge of the pages on the front.*** *More space is allowed that we think should commonly be needed to answer each question so that everyone has adequate space.* ***Please write with sufficient contrast for the scanner to be able to detect your writing.***

If you can't read your handwriting, neither can we.

1. *[6 points]*
(A) Here are the complete contents of file q1.c:

```
int *sub(void *a, int b);
int main(int argc, char *argv[]) {
     int sum = 0;
     return *sub(&sum, argc);
}
```

This file is compiled with the command:

```
gcc -std=c11 -Wall -g -c q1.c
```

(The `-c` switch tells gcc to compile but not link.)

Are any errors or warnings raised by the compiler? If so, explain briefly what they are complaining about.

(B) Now imagine that a file containing the implementation of `sub` is compiled and linked with the code above. No errors or warnings are raised in doing so.
Which of main's local variables (argc, argv, and sum) could possibly have a different value on return from sub than it had when sub was called?

(C) If sub contains no bugs, which of main's local variables (argc, argv, and sum) may have a different value on return from sub than it had when sub was called?

2. *[4 points]*
I compile my program and run it under `valgrind`. As part of its output, `valgrind` reports the following:

```
s
==6391== 100 bytes in 1 blocks are definitely lost in loss record 1 of 1
==6391==    at 0x4C2DB8F: malloc (in /usr/lib/valgrind/vgpreload_memcheck-amd64-linux.so)
==6391==    by 0x4005A7: sub3 (in /home/zahorjan/cse333/16au/midterm/code/q2)
==6391==    by 0x4005BF: sub2 (in /home/zahorjan/cse333/16au/midterm/code/q2)
==6391==    by 0x4005E5: sub1 (in /home/zahorjan/cse333/16au/midterm/code/q2)
==6391==    by 0x400614: main (in /home/zahorjan/cse333/16au/midterm/code/q2)
```

(A) Briefly explain what this output means.

(B) Briefly explain how `valgrind` obtained this information.
Hint: this could be a CSE 351 question.

3. *[3+2=5 points]*
(A) Write a C function that takes an array of integers and an integer array length as arguments and returns the largest element in the array.

(B) Modify you function so that it can also return a success/failure indication (in addition to the largest array element).

4. *[3 points]*
Here's an interface design for a dictionary implementation in C. What is seriously wrong with it? (Missing some method you'd like to have, but that isn't essential, isn't a serious failure.)

```
typedef void* dictionary_key_t;
typedef void* dictionary_value_t;
typedef struct dictionary_t *Dictionary;
Dictionary dictionaryCreate();
void dictionaryDestroy(Dictionary d);
void dictionaryAdd(Dictionary d, dictionary_key_t k,
                                        dictionary_value_t v);
dictionary_value_t dictionaryGet(Dictionary d, dictionary_key_t k);
```

5. *[4 points]*
Standard C provides two interfaces to the file system. One interface (we'll call it "interface A") has methods `open`, `read`, and `write`, while the second ("interface B") has `fopen`, `fread`, and `fwrite`.
(A) What is the **main** distinction between the two interfaces?

(B) Why does it make sense to have two interfaces? Why doesn't one of them dominate the other?

6. *[4 points]*
A user types the following command into a Linux shell (say, bash):

```
$ ./myprog | grep foo
```

Briefly explain what the bash code does to run the executables `./myprog` and `grep` and to connect them by an anonymous pipe.

7. *[3 points]*
Why are there both named pipes (also called fifo's) and anonymous pipes (as in the previous question)? In what circumstances are named pipes required, because anonymous pipes can't be used?

8. *[2 points]*
Write C code that, when run, will allocate 200 bytes or more of memory on the stack.

9. *[6 points]*
```
int myMemcmp(const void *s1, const void *s2, size_t n);
```
is a method that compares two regions of memory of size n bytes. It returns 0 if they are equal and -1 if not. Implement `myMemcmp`.

10. *[4 points]*

```
typedef  struct q11_t { int x; int y; } Q11;
```

Write a C method, `createQ11`, that returns a pointer to a new `Q11` that has both `x` and `y` set to 0.

11. *[12 points]*

Write an application that counts the number of times each lower case letter appears twice in a row in a file. The double letter occurrences cannot overlap. For instance, if the file contains

```
aaabbaacbAAa
```

the output would be

```
Results:
aa 2
bb 1
```

(If you don't remember the name of some include file, make your best guess. I know that you know that `man` will tell you the name, so having the name completely right isn't a grading criterion.)

Optional bonus question *[2 points]*

You have a file containing only this:

```
main() { printf("bonus question\n"); }
```

You try to build and then run it. What happens? (Circle all that apply.)

(A) It won't compile because you haven't declared the return type for method main().
(B) It won't compile, saying that main should be int (*)(int, char *[])
(C) It won't compile because you don't return anything from main
(D) It won't compile because you forgot to #include <stdio.h>
(E) It compiles, but there are many, many warnings
(F) It compiles but there is only one real warning (about printf)
(G) It compiles with no warnings or errors
(H) It compiles but won't link, saying you have the type of main wrong
(I) It compiles but doesn't link, saying there is no main that it recognizes
(J) It compiles and links but immediately crashes when you run it because you forgot to #include <stdio.h>
(K) It compiles and links but never terminates when it's run because you forgot to return anything from main
(J) It compiles and links and then prints "bonus question\n" and terminates when run