CSE 333 – Winter 2017
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. *[3 points]*
What happens when you try to compile, link, and run the code below? (Circle one.)

(A) It gets a compile time error
(B) It gets a link time error
(C) It gets a run time error
(D) It runs just fine

```
 #include <stdio.h>
#include <stdlib.h>

typedef struct s_t {
  int x;
  int y;
} * Widget;

Widget Widget_make() {
  Widget w = {0, 0}; malloc(sizeof(*w));
  if ( w == NULL ) return NULL;
  w->x = w->y = 0;
  return w;
}

int main(int argc, char *argv[]) {
  Widget W = Widget_make();
  printf("{%d, %d}\n", W.x, W.y);  W->x, W->y );
  free(W);
  return EXIT_SUCCESS;
}
```

2. *[8 points]*
This question reuses the code from the previous question **except that** type Widget is `typedef`'ed to a **pointer** to a `struct s_t`.
Indicate on the code above all the changes required (including any changes required to fix any issues already present in the code as written above) to get this modified version to compile , link, run, and produce output "{0,0}" with a clean Valgrind report

3. *[3 points]*
Suppose file `q3-foo.h` contains this:

```
#ifndef FOO_H
#define FOO_H
int foo() { return 0; }
#endif // FOO_H
```

and file `q3-sub.c` contains this:

```
#include "q3-foo.h"
```

```
int mysub() { return foo(); }
```

and file `q3-main.c` contains this:

```
#include "q3-foo.h"
int mysub();
int main(int argc, char *argv[]) {
  int answer = mysub();
  return answer;
}
```

You try to build using the command
`gcc -std=c11 -g -Wall q3-main.c q3-sub.c`
and if the code builds you try to run it.
Explain what happens during the build process, or what value is returned from main if it builds and is executed.

A function is defined in q3-foo.h, and q3-foo.h is #include'd in two different source files. When you try to link, you get an error about multiple definitions of "foo."

4. *[4 points]*
Design and implement a function that determine whether or not a byte range in memory reads the same backwards and forwards. For instance, if the byte range contains these bytes (given in hex)
00 0C C0 0C 00
your function should indicate the range reads the same forward and backwards. If the range were
00 0C C0 D0 E0 E0 D0 FF
it would not read the same forward and backwards.

Your must decide on the function's arguments and return value type.
The function does not need to check for errors, nor to return any indication of success or failure.

```
// returns true for a symmetric byte range, false otherwise
int byteSymmetric(char *p, int len) {
  for (char *end = p + len - 1; p<=end; p++, end--) {
    if ( *p != *end ) return 0;
  }
  return 1;
}
```

5. *[2 points]*
Suppose the line below declares a local variable. How many bytes of stack space will be allocated at run time because of this line on each entry to the function it is in?

```
char *str = "initial string";
```

8 (or 4)

6. *[4 points]*
Consider this C code fragment that declares two struct variables and then does an assignment:

```
#include <stdlib.h>

struct sone_t {
  int x;
  int y;
} S1;
struct stwo_t {
  char *str1;
  char* str2;
} S2;

int main(int argc, char *argv[]) {
  S1.x = S1.y = -1;
  S2.str1 = S2.str2 = NULL;
  S1 = *(struct sone_t*)&S2'
  return EXIT_SUCCESS;
}
```

(A) What are the values of S1 and S2 when the return statement is reached?

S1 = {0,0}
S2 = {NULL, NULL}

(B) Will Valgrind report any memory access errors if run on this code? Briefly justify your answer.

No. S1 is no bigger than S2, so casting S2 to be the size of S1 then copying bytes won't touch any bytes not part of S2 (and all of S2 has been initialized).

7. *[5 points]*

T	F	In C, all arguments are passed by value.
T	F	In C, all function return values are returned by value.
T	**F**	In C, if variable p is of type char*, then the value of *p is a C string.
T	**F**	If p is of type char* and p is not NULL, then *p = 'A'; cannot fail at run time.
T	**F**	If p is a void***, then p = &&&p sets p to its own address.
T	F	The code fragment p="foo"; *p='m'; is always an error (no matter what other code there might be around it)
T	**F**	You can't create a pointer to a function pointer in C
T	F	Type void* is often used in C to mean "any type"
T	**F**	The number of entries in the vtable for a "class" is the sum of the number of entries in the vtables of its ancestor classes plus the number

of methods it itself implements.

T F An application (consisting of many source files) can't have two functions both called `max`.

8. *[2 points]*
Suppose your C program tries to open a file, and suppose the file named in the open call exists, and suppose the open fails. Give an example reason why it might have failed.

Permission violation.

9. *[2 points]*
Suppose `main.c` contains a declaration of a global integer

```
int X;
```

File `sub.c` does not #include any files, and the only mention of X in it is this use:

```
int localX = X;
```

If you try to compile and link `main.c` and `sub.c`, what happens (based only on the lines of code I've shown here)?

You get an undeclared symbol error at compile time of sub.c.

10. *[4 points]*
This code compiles and links. What does it print when run?

```
#include <stdio.h>
char *sub(char * p, char c) {
  for (p=p; *p && *p!=c; p++);
  return p;
}
int main(int argc, char *argv[]) {
  char p[] = "abcdefghijklmnopqrstuvwxyz";
  printf("%lu\n", sizeof(p));
  printf("%s\n", sub(p, 't'));
  return 0;
}
```

27
tuvwxyz

11. *[10 points]*
Implement a module, Max, that takes remembers the largest item it has been given, and can give that largest item back. Max should be generic – the client code can hand it anything. In standard use, the client code would hand it one item after another and then ask for the largest one. The type of the items being handed over is known only to the client code.

Implement files Max.h, Max.c, and main.c, where main.c just shows some simple example use of your module. (main.c does not have to exhaustively test your code. It just has to

create a max “instance”, hand it some values, and get the max of those values out of it.)

Your module must be generic.

qmax.h

```
#ifndef QMAX_H
#define QMAX_H

// returns 1 for a<b, 0 otherwise
typedef int (*max_less_fn)(void *a, void *b);

typedef struct {
  max_less_fn less;
  void *      data;
} Max;

void  Max_init(Max *m, max_less_fn fn);
void *Max_destroy(Max *m); // returns stored item
void *Max_get(Max *m);
void *Max_put(Max *m, void *val); // returns value not stored

#endif // QMAX_H
```

qmax.c

```
#include "qmax.h"

void  Max_init(Max *m, max_less_fn fn) {
  m->less = fn;
  m->haveVal = 0;
}
void *Max_destroy(Max *m) {
  return m→data;
}
void *Max_get(Max *m) {
  return m->data;
}
void *Max_put(Max *m, void *val) {
  void *returnVal = m->data;
  if ( m->less(m->data, val) ) {
    m->data = val;
  } else {
    returnVal = val;
  }
  return returnVal;
```

```
}
```

main.c

```
#include <stdio.h>
#include <stdint.h>

#include "qmax.h"

int myless(void *a, void *b) {
  if ( (intptr_t)a < (intptr_t)b ) return 1;
  return 0;
}

int main(int argc, char *argv[]) {
  Max m;
  Max_init(&m, myless);
  Max_put(&m,  (void*)1);
  printf("%ld\n", (intptr_t)Max_get(&m));
  Max_put(&m, (void*)2);
  printf("%ld\n", (intptr_t)Max_get(&m));
  Max_put(&m, (void*)-1);
  printf("%ld\n", (intptr_t)Max_get(&m));
  Max_destroy(&m);
  return 0;
}
```