University of Washington – Computer Science & Engineering
Winter 2020 Instructor: Justin Hsia 2020-02-14

CSE333 MIDTERM

Last Name: Perfect
First Name: Perry
Student ID Number: 1234567
Name of person to your Left | Right Samantha Student | Larry Learner

All work is my own. I had no prior knowledge of the exam contents nor will I share the contents with others in CSE333 who haven’t taken it yet. Violation of these terms could result in a failing grade. (please sign)

Do not turn the page until 5:00.

Instructions

• This exam contains 10 pages, including this cover page. Show scratch work for partial credit, but put your final answers in the boxes and blanks provided.
• The last page is a reference sheet. Please detach it from the rest of the exam.
• The exam is closed book (no laptops, tablets, wearable devices, or calculators). You are allowed one page (US letter, double-sided) of handwritten notes.
• Please silence and put away all cell phones and other mobile or noise-making devices. Remove all hats, headphones, and watches.
• You have 70 minutes to complete this exam.

Advice

• Read questions carefully before starting. Skip questions that are taking a long time.
• Read all questions first and start where you feel the most confident.
• Relax. You are here to learn.

<table>
<thead>
<tr>
<th>Question</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Possible Points</td>
<td>19</td>
<td>10</td>
<td>24</td>
<td>32</td>
<td>19</td>
<td>104</td>
</tr>
</tbody>
</table>
**Question 1:** You MAKE Me Whole  [19 pts]

Let `CFLAGS = -Wall -g -std=c11`. The symbol “$^” means all sources.

(A) Complete the corresponding directed acyclic graph for the Makefile.  [5 pt]

(B) Starting with only the source files (.c and .h) and Makefile, what should happen to the following files if we run “make” followed by “make clean”? Use “C” for created, “CD” for created and then deleted, and “U” for untouched (i.e. unchanged or not created).  [4 pt]

(C) Do we need a phony all target in Makefile? Briefly justify your response.  [2 pt]


Follow the DAG from rain.h to see that rain.o and snow.o are affected, but clouds.o is not. Since we ran the winter target, we don’t attempt to rebuild snow.
Assuming that the two executables do different things, it turns out that there is something inherently wrong with our project setup that will cause 1 of 2 possible compilation errors. Identify the compilation errors and which target will cause them. **Hint:** what does *every* C executable need? [4 pt]

<table>
<thead>
<tr>
<th>Possible error</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>redefinition of main</td>
<td>winter</td>
</tr>
<tr>
<td>missing main</td>
<td>snow</td>
</tr>
</tbody>
</table>

Every executable needs a main function. *winter* and *snow* must have different mains in order to do different things. *snow.o* is linked into both executables, so either it (and *snow*) doesn’t have code for main, or two mains are linked into *winter*.

<table>
<thead>
<tr>
<th>Possible error</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>missing symbols/functions from <em>rain.o</em> and <em>clouds.o</em></td>
<td>snow</td>
</tr>
</tbody>
</table>

The *snow.o* target includes *rain.h* and *clouds.h*, but the *snow* target only includes *snow.o*, meaning that it would be missing the implementations/definitions of anything it needs from those interfaces.
**Question 2: PREPROCESS This! [10 pts]**

Suppose we have the following files:

**twoface.h:**
```c
#ifndef DSWITCH
#define FACE(f) NULL
#else
#define FACE(f) (f * -2)
#endif
typedef int my_type;
```

**twoface.c:**
```c
#include <stdio.h>
#define f 2.0
#include "twoface.h"
int main(int argc, char** argv) {
    printf("%ld\n", (long) FACE(f) );
    return 0; // EXIT_SUCCESS
}
```

(A) The header file is missing a header guard! Following the style guide for this class, what name should we use for the guard macro? [2 pt]

(B) Complete the result of `cpp -P -DSWITCH twoface.c` below. Ignore the output of the `#include <stdio.h>` directive. [5 pt]

```c
typedef int my_type;
int main(int argc, char **argv) {
    printf("%ld\n", (long) (2.0 * -2) );
    return 0;
}
```

We have defined the symbol SWITCH in the command-line arguments to cpp, meaning DSWITCH is not defined. The preprocessor also removes all comments.

(C) (Circle one) What will happen when we try to compile `gcc -DSWITCH twoface.c` and run `a.out`? [3 pt]

<table>
<thead>
<tr>
<th>compiler error</th>
<th>output</th>
<th>output</th>
<th>output</th>
</tr>
</thead>
<tbody>
<tr>
<td>compiler error</td>
<td>-4</td>
<td>0</td>
<td>4</td>
</tr>
</tbody>
</table>

The syntax is good, so will compile and execute as expected.
Question 3: ORDER Up  [24 pts]

We’re writing C software for restaurants to track orders using the following typedef-ed struct:

```c
#define NUM_MENU_ITEMS 3
typedef struct order_st {
    int table;  // table number
    char* server;  // name of server
    int orders[NUM_MENU_ITEMS];  // # of each menu item ordered
    struct order_st* next;  // next order in linked list
} Order;
```

We use `Order* head` to track all orders and `Order* curr` to track the current order. Assume both are defined in `main`. Because we cannot predict how many orders we will get, `Orders` must be allocated individually on the heap.

(A) Draw a memory diagram for a small linked list of two orders. The first order is for table 3, served by "Andrew", and is for 1 of menu item #1. The second (and current) order is for table 7, served by "Cheng", and is for 2 of menu item #0 and 4 of menu item #2.

Character arrays can be written as string literals. Don’t forget to include variable and field names. [8 pt]
(B) Below, complete the helper function `CreateOrder()` that generates a new, empty order (i.e., 0 quantity of all menu items) with some specified field values. Assume that `*server` doesn’t need to be deep-copied. `NUM_MENU_ITEMS` is `#define d`. [8 pt]

```c
// Returns a pointer to an empty order, or NULL on error.
Order* CreateOrder(int table, char* server) {
    Order* order = (Order*) malloc(sizeof(Order));
    if (order != NULL) {
        order->table = table;
        order->server = server;
        for (int i = 0; i < NUM_MENU_ITEMS; i++) {
            order->orders[i] = 0;
        }
        order->next = NULL;
    }
    return order;
}
```

(C) Recall that `head` and `curr` are local pointers in `main`. We are writing `AddOrder` that takes a specified heap-allocated `Order` (e.g., the return value from `CreateOrder`) and adds it to the end of the `head` list. If either `head` or `curr` is `NULL`, then they need to be updated to point to this new `Order`, meaning we may need to update the values of both `head` and `curr` in this function. Following good style guidelines, propose a suitable declaration: [4 pt]

Google preferred: `Order* AddOrder(Order* new_order, Order** curr);`

Acceptet: `void AddOrder(Order* new, Order** head, Order** curr);`

Return values preferred to output params; input params come before output params.

(D) If we want to create a module for our `Order` system, indicate which file the following would go in (checkmark): [4 pt]

<table>
<thead>
<tr>
<th></th>
<th>Order.h</th>
<th>Order.c</th>
<th>Restaurant.c</th>
</tr>
</thead>
<tbody>
<tr>
<td>Order typedef from problem description</td>
<td>✔️</td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>CreateOrder()</code> definition from part B</td>
<td></td>
<td>✔️</td>
<td></td>
</tr>
<tr>
<td><code>CreateOrder()</code> declaration</td>
<td>✔️</td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>main()</code></td>
<td>✔️</td>
<td></td>
<td>✔️</td>
</tr>
</tbody>
</table>
Question 4: Time to Get in SHAPE [32 pts]

Abbrev: constructor (ctor), copy constructor (cctor), assignment (op=), destructor (dtor).

```cpp
struct Point {
    Point() : x(0), y(0) { }
    Point(int x, int y) : x(x), y(y) { }
    int x, y;
};  // struct Point

class Shape {
    public:
        Shape() : num_pts_(1), points_(new Point) { }
        Shape(const Shape& s);  // DEEP copies data members
        Shape& operator=(const Shape& rhs);  // DEEP copies
        ...  // other methods mentioned in this question
    private:
        Point* points_;  // array of num_pts_ points [Heap]
        size_t num_pts_;  // # of points in shape
        uint8_t color[3];  // RGB values of shape color
    };  // class Shape
```

(A) Do we need accessor methods for `Point`? Briefly explain why or why not. [2 pt]

```
No, because the data members in `Point` are publicly-accessible by default.
```

(B) Write out a line of code that will disable the cctor inside the definition `Point`. [2 pt]

```
Point(const Point& p) = delete;
```

(C) What does a default `Shape` describe? [2 pt]

```
A point at the origin (0, 0) with random/garbage color.
```

(D) The member function `Area` returns the area of the `Shape` as a `double`. Propose a suitable function signature (for the `implementation` file): [3 pt]

```
double Shape::Area() const {
```

(E) The member function `ChangeColor` sets the `Shape`’s color to specified red, green, and blue values. Propose a suitable function signature (for the `implementation` file): [3 pt]

```
void Shape::ChangeColor(const uint8_t red, const uint8_t green,
                           const uint8_t blue) {
```
points_ points to an array on the heap. Define a Shape member function `Union()` that *appends* the points from a second Shape to `points_` in this. Don’t worry about duplicate points or self-unions. [10 pt]

```cpp
void Shape::Union(const Shape& s) {
    Point *old = points_; // def ctor
    points_ = new Point[num_pts_ + s.num_pts_];
    for (size_t i = 0; i < num_pts_; i++) // copy old
        num_pts_[i] = old[i];
    for (size_t j = 0; j < s.num_pts_; j++) // append new
        num_pts_[num_pts_ + j] = s.points_[j];
    num_pts_ += s.num_pts_; // increase size
    delete[] old; // deallocate old
} // many valid solutions exist
```

(G) The inline definition of the Shape destructor is given below, but leads to a memory error in our code! Briefly describe the issue and the fix (which may not be in the dtor): [4 pt]

```cpp
~Shape() { delete[] points_; }
```

<table>
<thead>
<tr>
<th>Issue</th>
<th>Fix</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mismatched delete for a default-constructed Shape (i.e., delete[] on Point).</td>
<td>Update def ctor initializer list to use points_(new Point[1]) instead.</td>
</tr>
</tbody>
</table>

(H) Assume that the Shape cctor (definition not shown) does a *deep* copy of data members. If `s` is a Shape with 2 points, how many times are each of the following invoked (count both Shape and Point methods) during the execution of the friend non-member function `Reverse(s)`? [6 pt]

```cpp
Shape Reverse(const Shape& s) {
    Shape out = s;
    for (size_t i = 0; i < s.num_pts_; i++) {
        out.points_[i] = s.points_[s.num_pts_ - 1 - i];
    }
    return out;
}
```

`ctor` _4_  `cctor` _2_  `op=` _6_  `dtor` _3_

`ctor` of Shape is called twice (out, return), each time calling 2 def cctor & 2 op= of Point. There are an additional 2 op= of Point in the for-loop. The Shape dtor is called on out after Reverse returns, which also destructs the 2 Points in its array.
**Question 5: INPUT and OUTPUT and ERRORS, oh my!** [19 pts]

(A) Assume that the C std lib is using an internal write buffer of **1024 bytes** and we are trying to write **2048 bytes** total in **256-byte chunks**. Assuming that all writes are successful (i.e. no partial writes or errors), how many system calls do we invoke using C std lib vs. POSIX? [4 pt]

<table>
<thead>
<tr>
<th>write()</th>
<th>fwrite()</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>2</td>
</tr>
</tbody>
</table>

write() will invoke a system call every time (2048/256 = 8 times). fwrite() will only invoke a system call when it flushes its buffer (2048/1024 = 2 times).

(B) Name a C function that we have used in this class that fits the descriptions: [4 pt]

- Part of the C standard library, but doesn’t invoke a system call. *e.g.,* strncpy, sqrt
- A POSIX system call that doesn’t have a C std lib equivalent. *e.g.,* opendir

(C) Convert the following two lines of C code into their C standard library equivalents. Do NOT add any other lines (e.g. error checking): [5 pt]

**POSIX:**
```c
int fd = open("midterm.txt", O_RDONLY);
ssize_t n = read(fd, buf, 333*sizeof(int32_t));
```

**C Std Lib:**
```c
_FILE* file = fopen("midterm.txt", "r");
size_t n = fread(buf, sizeof(int32_t), 333, file);
```

(D) Before exiting/terminating a C program, name the three categories of **resources** that we have seen in this class that we need to make sure are cleaned up/closed: [3 pt]

<table>
<thead>
<tr>
<th>dynamically-allocated memory</th>
<th>files / streams</th>
<th>directories</th>
</tr>
</thead>
</table>

(E) **Briefly** describe in what situations you prefer to use perror instead of fprintf to stderr. [3 pt]

When there are multiple possible causes of the error, as perror will print out an additional message related to errno. If there is a single cause of error, then a helpful fprintf message will suffice.