CSE 333 Midterm Exam Cinco de Mayo, 2017 (May 5)

Name _____ UW ID# _____

There are 6 questions worth a total of 100 points. Please budget your time so you get to all of the questions. Keep your answers brief and to the point.

The exam is closed book, closed notes, closed electronics, closed telepathy, open mind.

If you don't remember the exact syntax for something, make the best attempt you can. We will make allowances when grading.

Don't be alarmed if there seems to be more space than is needed for your answers – we tried to include more than enough blank space.

Relax, you are here to learn.

Please wait to turn the page until everyone is told to begin.

Score _____ / 100

1. _____ / 12

2. _____ / 20

3. _____ / 20

4. _____ / 15

5. _____ / 31

6. _____ / 2

Question 1. (12 points) Preprocessor. Suppose we have the following two files:

```
hdr.h:
   #define XVII 17
   #define QUAD(a,b,c,x) (a*x+b)*x+c
   int fun(int a, int b);

main.c:
   #include <stdio.h>
   #include "hdr.h"
   int main() {
      int n = XVII+1;
      int k = QUAD(1+2, 4, XVII, 3-2);
      printf("n = %d, k = %d, XVII = %d\n", n, k, XVII);
      return 0;
   }
```

(a) (8 points) Show the result produced by the C preprocessor when it processes file `main.c` (i.e., if we were compiling this file, what output would the preprocessor send to the C compiler that actually translates the program to machine code?). You should ignore the `#include <stdio.h>` directive since that includes library declarations that we do not have access to. Write the rest of the preprocessor output below.

(b) (4 points) What output does this program print when it is executed? (It does compile and execute without errors.)

Question 2. (20 points) C data structures. The following code defines data structures to describe a colored point in 2-D space, and includes a partly-written function to allocate a new color point as well as the key parts of a `main` function that would use it. Your job is to complete the implementation of the function, and draw a diagram of memory right before this function returns to where it was called from `main`.

Here is the source code for everything except the `NewColorPoint` function, which is on the next page along with space for the answers.

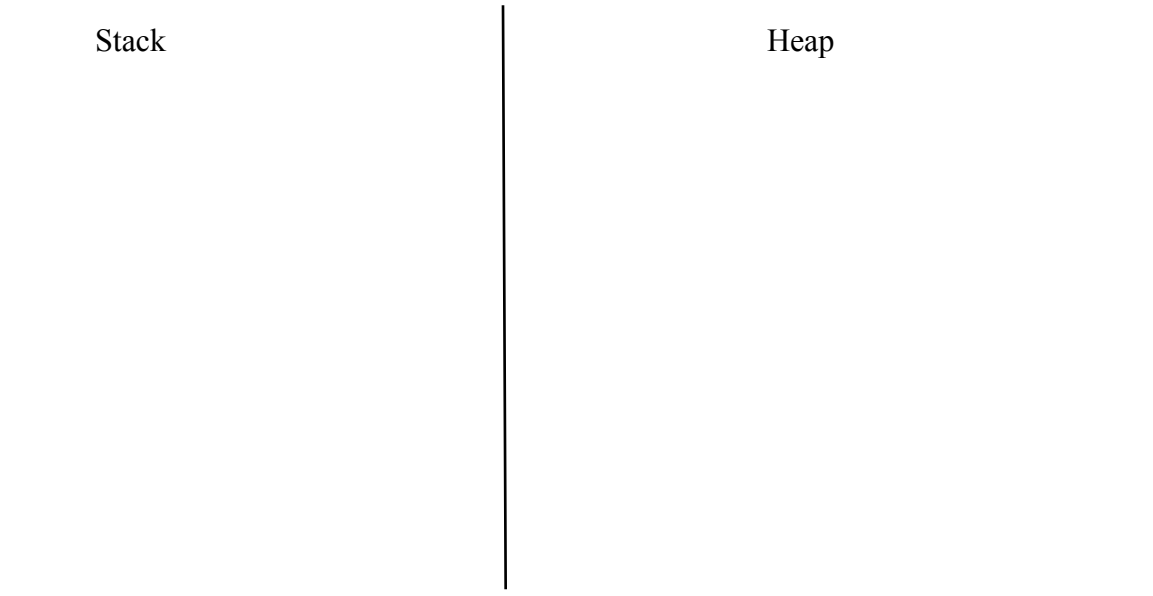
```
#include <stdlib.h>
#include <stdint.h>

// 2-D point
typedef struct point_st {
  int x, y;    // x and y coordinates
} Point2D;

// Color 2-D point
typedef struct cpoint_st {
  Point2D pt;         // coordinates of this point
  uint8_t rgb[3];     // red/green/blue color values
                      //   (0-255 each, in the order r,g,b)
} ColorPoint, *ColorPointp;


// function NewColorPoint is on the next page of the test

// main program - call NewColorPoint to allocate a ColorPoint
// and free it when done
int main() {
  ColorPoint * cpt = NewColorPoint(3,4,5,0,111);

  // do stuff with cpt....

  free(cpt);
  return 0;
}
```

(Continued on the next page. You can remove this page from the exam if you wish.)

Question 2. (cont.) (a) (10 points) Complete the implementation of function `NewColorPoint` by adding code in the spaces below. These include the function result type, the type of variable cp, the missing parts of the malloc function call, the body of the function to initialize the `ColorPoint`, and the return expression at the end.

```
// Allocate a ColorPoint on the heap, initialize it with given
// (x,y)coordinates and (r,g,b) color values, and return pointer
// to newly allocated & initialized ColorPoint.

____________________ NewColorPoint(int x, int y,
                                   uint8_t r, uint8_t g, uint8_t b) {

  _____________ cp = (_________________)malloc(____________________);
  // additional code below









  // HERE!!
  return ____________________;
}
```

(b) (10 points) Draw a diagram that shows the contents of memory when execution reaches the point marked `HERE!!` in `NewColorPoint`, after it is called by `main` as shown on the previous page. Be sure to show the contents of the stack frames for `main` and `NewColorPoint`, as well as heap storage, and draw all appropriate pointers.



Question 3. (20 points) More C programming. This question involves adding a function to the code from HW1. Copies of the `LinkedList.h` and `LinkedList_priv.h` header files, which are the only ones that might be needed for this queston, are provided on separate pages. You should assume that those headers, and any other necessary header files and declarations, have already been included in the source files.

Several people have suggested it would be very helpful if there was an additional function in the `LinkedList` package that would delete a payload from a list if it is found there. If there is more than one copy of the payload in the list then the first copy should be removed. If the payload is not found in the list then the list remains unchanged and the function returns a result to indicate that the remove operation failed.

Here is the prototype and specification for the new function:

```
// Removes the first occurrence of the payload.
//
// Arguments:
//
// - list: the list to operate on
//
// - payload: the payload to remove
//
// Returns false on failure, true on success
bool LLRemoveFirstOccurence(LinkedList list,
                            LLPayload_t payload);
```

Complete the implementation of this function on the next page. For full credit, your solution must use other functions in the `LinkedList` package when appropriate and not duplicate code that is already implemented elsewhere. (Specifically, you should use an iterator and related functions to process the list.) You may assume that the following function has already been defined for you to use if needed:

```
static void LLNullFree(LLPayload_t freeme) { }
```

(Hint: Don't panic! The solution isn't particularly long – the sample solution is about a dozen lines of code. Of course, yours isn't required to be any particular length. You may remove this page from the exam if you wish.)

…

…

…

…

Question 3. (cont.) Complete the implementation of `LLRemoveFirstOccurrence` below.

```
bool LLRemoveFirstOccurence(LinkedList list,
                            LLPayload_t payload) {
  LLIter it = LLMakeIterator(list, 0);
  if (it == NULL) {
    return false;
  }

  // Add your code below









}
```

Question 4. (15 points) We'll describe what it prints, you show us how!

Consider the following C++ code (not plain C), which has `???` in the place of function names in three places in `main`:

```
struct Thing {
  int a;
  bool b;
};

void PrintThing(Thing t) {
  cout << "Thing:" << endl;
  string val = t.b ? "true" : "false";
  cout << "\t" << t.a << ", " << val << endl;
}

void f1(Thing t);
void f2(Thing &t);
void f3(Thing *t);
void f4(const Thing &t);
void f5(const Thing t);

int main() {
  Thing foo;
  foo.a = 5;
  foo.b = true;
  PrintThing(foo);

  cout << "\nFirst mystery: " << endl;
  __???__(foo);
  PrintThing(foo);

  cout << "\nSecond mystery: " << endl;
  __???__(&foo);
  PrintThing(foo);

  cout << "\nThird mystery: " << endl;
  __???__(foo);
  PrintThing(foo);

  return 0;
}
```

(continued on next page – you can remove this page for reference.)

Question 4. (cont.) (a) We're trying to reverse-engineer this program to figure out which of functions `f1` through `f5` might have been called at the three points labeled with `???` in the original program. We don't know anything about the code in those functions, but we do have a printout showing what happened when the program was executed:

```
Thing:
     5, true

First mystery:
Thing:
     6, false

Second mystery:
Thing:
     3, true

Third mystery:
Thing:
     3, true
```

Your job is to list all of the functions that *might* have been called at each of the three mystery points in the program. For each of these points, give the names of all of the functions that would compile cleanly (no errors) and *could have* produced the results shown. There is at least one possible function that could be called at each point to produce the given results; there might be more. (Hint: look at parameter lists and types in the function declarations and in the calls.)

First mystery: _____

Second mystery: _____

Third mystery: _____

Question 5. (31 points) Cinco de Mayo in C++ !! In honor of the occasion, we would like to implement simulated burritos. The following class describes a Burrito, but it is missing some code. Many of the functions print extra messages to `cout` so we can trace execution of code that uses the class.

```
#include <iostream>
using namespace std;

class Burrito {
 public:

  // Construct bland Burrito of unknown kind
  Burrito(): kind_("???"), spicy_(0) {
    cout << "default ctr" << endl;
  }

  // Construct Burrito with given kind and spicy value
  Burrito(string kind, int spicy)
          : kind_(kind), spicy_(spicy) {
    cout << "constructor" << endl;
  }

  // copy constructor - initialize this to be a copy of b
  Burrito(const Burrito &b);

  // destructor
  ~Burrito() { cout << "destructor" << endl; }

  // accessors
  string get_kind()  const { return kind_; }
  int    get_spicy() const { return spicy_; }

 private:
  // Burrito representation
  string kind_;   // kind ("cheese", "bean", etc.)
  int spicy_;     // how hot (0=bland, 1=mild, 2=hot, etc.)
};
```

Answer questions about this class on the following pages. You can remove this page from the test if you wish.

Question 5. (cont.) (a) (5 points) Complete the implementation of the copy constructor for this class. This code is intended to be in a separate implementation (.cc) file from the header file containing the class declaration. It should write the string "copy ctr" to `cout` when it is executed (provided below). (The answer to this question might be quite short.)

```
Burrito::Burrito(const Burrito &b) {
  cout << "copy ctr" << endl;
  // add additional code below as needed




}
```

(b) (6 points) We would like to add an assignment operator to this class so that `b1=b2;` copies the contents of `Burrito b2` to `b1`. Write an appropriate declaration for the assignment operator that we should add to the class declaration for `Burrito`.

(c) (10 points) Write an implementation (definition) of the assignment operator declared in part (b). This code is also intended to be placed in a.cc file that is separate from the header file, just as with the copy constructor. Your implementation should write the string "assign" to `cout` when it is executed. You should include all of the code that belongs in a proper C++ `operator=`, even if some of it isn't strictly needed in this case.

(continued on next page)

Question 5. (cont.) (d) (10 points) Now suppose we execute the following program using the `Burrito` class with the copy constructor and assignment operations implemented in the previous parts of this question. What output is produced when we run this program? (Most of the output is printed by the various member functions when they are called.)

```
int main() {
  Burrito *veggie = new Burrito("bean", 3);
  Burrito another = *veggie;
  cout << "kind = " << another.get_kind() << endl;
  Burrito more;
  more = another;
  delete veggie;
  cout << "spicy = " << more.get_spicy() << endl;
  return 0;
}
```

Output:

Question 6. (2 free points) (All reasonable answers receive the points. All answers are reasonable as long as there is an answer. ☺)

(a) (1 point) What question were you expecting to appear on this exam that wasn't included?

(b) (1 point) Should we include that question on the final exam? (circle or fill in)

Yes

No

Heck No!!

$!@$^*% No !!!!!

None of the above. My answer is _____.