CS16—Final Exam

E03, W15, Phill Conrad, UC Santa Barbara

Wednesday, 03/17/2015, 7pm-10pm

Name: _____

Umail Address: _____@ umail.ucsb.edu

- Please write your name **above AND AT THE TOP OF EVERY PAGE**
- Be sure you turn in every page of this exam.
  - Each exam is numbered (e.g. Exam #137).
  - Each pages is numbered (e.g. Page 1, Page 2, etc.)
  - The last page clearly says "End of Exam".
- This exam is **closed book, closed notes**, **closed mouth, cell phone off**
- You are permitted **one sheet of paper** (max size 8.5x11") on which to write notes
- These sheets will be collected with the exam, and might not be returned
- Please write your name on your notes sheet

1. a. (3 pts) Convert 8415 from hexadecimal to base 2 1000 0100 0001 0101

   b. (3 pts) Convert 57 from octal to base 2 101 111

   c. (3 pts) Convert 0100 1001 0100 1010 from base 2 to hexadecimal 494a

2. For these questions, assume 8-bit two's complement representation of negative integers. The left-most bit is the sign bit, with 1 representing negative numbers, and 0 representing positive numbers.

   a. (3 pts) Given the decimal number -91, what is this number's binary representation in 8-bit two's complement?

   10100101

   b. (3 pts) Given that 10111101 is the 8-bit two's complement representation of a number, what is that number in base ten?

   -67

3. Fill in the incomplete parts of the function definitions below and on the next page. These functions use structs defined in `recipe.h` which is on a separate handout given with this exam.

   You should find parts (a) and (b) on this page.

   Parts (c), (d), (e) and (f) are on the next page, but this is ALL ONE PROGRAM, and ALL ONE QUESTION.

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

#include "recipe.h"

double totalCalories(Recipe r) {
  double result = 0.0;

  // (a) (4 pts) Add a line of code here that declares the variable n with an
  //   appropriate type for how it is used in the for loop below.



  for (n=r.head; n!=NULL; n=n->next) {
    // (b) (4 pts) Add a line of code here that accumulates the total calories
    //  in the recipe.   Note that for each item in the linked list,
    //  the calories for that item is the result of multiplying
    //  cups times calsPerCup, so it is that product that must be added
    //  to the total.



  }
  return result;
}
```

CONTINUED ON NEXT PAGE

CONTINUED FROM PREVIOUS PAGE

```
// Add Ingredient to a Recipe
void addToRecipe(Recipe *r, Ingredient i) {

  // Declare a variable n of type pointer to IngrNode, and
  // initialize it to a new IngrNode struct on the heap (c) (4 pts)



  n->ing = i;
  n->next = NULL;

  // FILL IN the boolean condition for the if test? (d) (4 pts)

  if (                                          ) {
    r -> head = n;
    r -> tail = n;
  } else {
    r -> tail -> next = n;

    // FILL IN A LINE OF CODE HERE  (e) (4 pts)



  }
}

int main() {
  Ingredient ingredients[] = {
    {"dry steel cut oats", 0.67, 600},
    {"soy milk", 1.67, 110},
    {"maple syrup", 0.125, 840},
    {"vanilla extract", 0.02, 599}, // 1 tsp
    {"cinnamon", 0.02, 288}, // 1 tsp
    {"chopped dates", 0.5, 532},
  };
  Recipe creamyOats = { NULL, NULL};

  for (int i=0; i<6; i++ ) {

    // (f) (4 pts) Fill in a single line of code here that calls the function
    //  addToRecipe each time through the loop to that each element
    //  of ingredients gets added to the Recipe creamyOats.



  }
  double calories  = totalCalories(creamyOats);
  cout << "Total Calories=" << calories << endl;
  return 0;
}
```

4. Most of the code we wrote in this course goes in files that end in .cpp, for example: `arrayFuncs.cpp`, or `arrayFuncTests.cpp`.

   But sometimes we worked with each of the following types of files. For each type, explain what its purpose is, and its relationship to .cpp files.

   DON'T USE TOO MANY WORDS. Write just enough to get the main point across. Excessively verbose answers that stray off point may be penalized.

   a. (2 pts) Purpose of .h files:

   b. (2 pts) .h files relationship to .cpp files:

   c. (2 pts) Purpose of .o files:

   d. (2 pts) .o files relationship to .cpp files.

   e. (2 pts) Purpose of Makefile:

   f. (2 pts) Relationship of Makefile to .cpp files:

5. Given the following declarations:

```
struct Node {
  int data;
  Node *next;
};


int main(int argc, char *argv[]) {
  Node x;
  double y;
  int z;
  char a;
  Node *b;
  double *c;
  int *d;
  char *e;


  return 0;
}
```

Specify the type of each of these expressions (e.g. `int`, `int *`, etc.

a. (3 pts) &c `double **`

b. (3 pts) argv[0] `char *`

c. (3 pts) *e `char`

d. (3 pts) b->next->next `Node *`

e. (3 pts) z `int`

f. (3 pts) b->next `Node *`

g. (3 pts) argc `int`

h. (3 pts) b->data `int`

i. (3 pts) argv[1][2] `char`

j. (3 pts) d `int *`

k. (3 pts) &a `char *`

6. Assume the `main` function in the program `runIt.cpp` starts with:

```
int main(int argc, char *argv[]) {
...
```

Further, suppose this program is invoked with the following command line:

```
./runIt lime lemon
```

a. (2 pts) What is the value of `argc` in this case? `3`

b. (2 pts) What is the value of argv[2][3]? `o`

c. (2 pts) What is the value of argv[1][0]? `l`

d. (2 pts) What is the value of argv[0][5]? `I`

7. (4 pts) Write a single line of code that declares an integer variable with the name `num1` and sets it equal to the value of `argv[1]`, converted to an integer.

You may *assume* that any necessary `#include` directives already have been done, so you don't need to worry about that. Just write the line of code.

8. (4 pts) Write a single line of code that declares a variable for a real number (one such as 3.45) with the name `num2` and sets it equal to the value of `argv[2]`, converted as necessary. The variable should have an appropriate data type.

You may *assume* that any necessary `#include` directives already have been done, so you don't need to worry about that. Just write the line of code.

End of Exam
total points=100