CS16—Final Exam

E03, F14, Phill Conrad, UC Santa Barbara

Wednesday, 12/15/2014

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. Please perform the following number conversions.

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

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

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

   d. (2 pts) Convert 75 from octal to base 2 111 101

   e. (2 pts) Convert 1110 0101 from binary to decimal 229

   f. (2 pts) Convert 0101 1010 from base 2 to decimal 90

   g. (2 pts) Convert 6086 from hexadecimal to base 2 0110 0000 1000 0110

   h. (2 pts) Convert 129 from decimal to base 2 1000 0001

   i. (2 pts) Convert 2fc6 from base 16 to base 2 0010 1111 1100 0110

   j. (2 pts) Convert 111 011 000 from base 2 to base 8 730

2. 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. (2 pts) &c `double **`

b. (2 pts) argv[0] `char *`

c. (2 pts) *e `char`

d. (2 pts) b->next->next `Node *`

e. (2 pts) z `int`

f. (2 pts) b->next `Node *`

g. (2 pts) argc `int`

h. (2 pts) b->data `int`

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

j. (2 pts) d `int *`

k. (2 pts) &a `char *`

3. (10 pts) In 24 hour time:

   - midnight is represented by the hour 0
   - 1am through 11am are represented at hour=1 through hour=11
   - noon is represented by hour=12
   - 1pm through 11pm are represetned by hour=13 through hour=23
   - minute is a value between 0 and 59.

   Assume the following struct:

```
// Time of day in 24 hour time
struct Time {
   int h; // hour
   int m; // minute
};
```

   Write the full function definition for a function that would have the following prototype:

```
  bool isValidTime(Time t);
```

   Comply with these instructions for full credit:

   - The function should return true if both of the members of the struct t are valid values for a struct Time. Otherwise the function should return false.
   - Assume that any `#include` directives you need are ALREADY DONE. I ONLY want the function definition. I want the full function definition, and NOTHING ELSE. PERIOD.

4. (10 pts) Given the same struct definition as in the previous problem, with the same understanding of how are the fields for hour and minutes are to be treated in 24 hour time:

```
// Time of day in 24 hour time
struct Time {
   int h; // hour
   int m; // minute
};
```

And given the same function prototype for `isValidTime`, which you can assume is ALREADY DEFINED and available for you to use (do not redefine it in this problem)

```
bool isValidTime(Time t);
```

Write the full function definition for a function that would have the following prototype:

```
bool setTime(Time *t, int hour, int min);
```

Comply with these instructions for full credit:

- If the pointer t is NULL, return false, and do nothing.
- If the pointer t is not NULL, copy the values from hour and min into the corresponding fields in the struct pointed to by t.
- Then, USING A FUNCTION CALL TO isValidTime, NOT BY REPEATED THE CODE YOU WROTE FOR THAT FUNCTION, check whether the struct you just initialized is valid. If so, return true, otherwise return false. Hint: be careful about matching the types of formal and actual parameters.

5. (10 pts) Given the struct definitions shown below (which are exactly as in lab09 and lab09), write the full function definition for a function that would have the following prototype:

```
int countOccurences(LinkedList * list, int k);
```

Comply with these instructions for full credit:

- The function should return 0 if `list` is NULL, or the the linked list that is pointed to is an empty list.
- Otherwise, the function should go through the linked list and count the number of nodes in the list that have the value `k` in the data member.
- Assume that any `#include` directives you need are ALREADY DONE. I ONLY want the function definition. I want the full function definition, and NOTHING ELSE. PERIOD.

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

struct LinkedList {
  Node *head;
  Node *tail;
};
```

6. (10 pts) Again, given the following struct definitions, exactly as in lab09 and lab09

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

struct LinkedList {
  Node *head;
  Node *tail;
};
```

Here is a partial implementation of addIntToEndOfList. It has one extra line of code added that will result in a memory leak, and it has one crucial line of code missing.

You have two jobs, and you must do this precisely as stated to get full credit:

- (5 pts) Add EXACTLY ONE LINE OF CODE, the missing line that will cause the function to work properly.
- (5 pts) Cross out EXACTLY ONE LINE OF CODE, the one that causes a memory leak.

HINT: Trace through the code with a picture, both starting from an empty list, and one that already has at least one node in it. Think about what you WANT the "after" picture to look like, vs. what the code you see here actualy does. Then you'll see both thing you need to fix.

NOTE: there may be 'other' ways to fix the code. I DON'T WANT THOSE. Fix it according to the instructions. Also, note that I've put exactly the same amount of space between each line, so that where the white space is is NOT a clue as to where the missing line of code should go.

```
void addIntToEndOfList(LinkedList *list, int value) {

 assert(list!=NULL); // if list is NULL, we can do nothing.

  Node *p;

  p = new Node;

  p->data = value;

  p->next = NULL;

  if (list->head == NULL) {

     list->head = new Node;

     list->head = p;

  } else {

     list->tail->next = p;

     list -> tail = p;

  }

}
```

7. 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

End of Exam

total points=100