CMPSC16 Final Exam
Spring 2017
06/12/2017

This exam is closed book, closed notes. However, you may use one 8.5" x 11" piece of paper with notes on both sides. Write your name on the your notes paper. No calculators or phones are allowed in the exam. **Write all your answers on the answer sheet. However, all exam sheets, notes paper and scratch paper must be turned in at the end of the exam.**

By signing your name below, you are asserting that all work on this exam is yours alone, and that you will not provide any information to anyone else taking the exam. In addition, you are agreeing that you will not discuss any part of this exam with anyone who is not currently taking the exam in this room until after the exam has been returned to you. This includes posting any information about this exam on Piazza or any other social media. Discussing any aspect of this exam with anyone outside of this room constitutes a violation of the academic integrity agreement for CMPSC16.

Signature: ____

Name (please print clearly):____

Umail address: ____________________________@umail.ucsb.edu

Perm number: ____

You have 3 hours to complete this exam. Work to maximize points. A hint for allocating your time: if a question is worth 10 points, spend no more than 10 minutes on it if a question is worth 20 points, spend no more than 20 minutes on it etc. If you don't know the answer to a problem, move on and come back later. Most importantly, stay calm. You can do this.

WRITE ALL YOUR ANSWERS IN THE PROVIDED ANSWER SHEET IN PEN!

DO NOT OPEN THIS EXAM UNTIL YOU ARE INSTRUCTED TO DO SO.
GOOD LUCK!

Name: ______________________

Part 1. [20 points] Multiple Choice: ***Select the SINGLE best answer by filling in the circles provided in your answer sheet. You must shade your selection completely as shown below:***

○ Not selected ● Selected

1. Which of the following is a valid declaration for a pointer to a 'struct Node'? Assume that the struct Node data type has been declared.

```
A. Node& p;
B. Node p;
C. Node* p;
D. Node** p;
E. None of the above
```

2. What is the output of the following code?

```
for (int i = 0; i < 3; i++){
     for (int j = 0; j < i; j++)
          cout<< i+j <<" ";
}
```

A. 0 2 4
B. 1 2 3
C. 0 1 2 3
D. 0 1 2 1 2 3 2 3 4
E. None of the above

3. Consider a function named `getAverage()` that RETURNS the average value in an integer array (`arr`) of length (`len`). Which of the following is a valid declaration of `getAverage()`?

```
A. void getAverage(int arr[], int len);
B. void getAverage(int arr[len]);
C. Void getAverage(int arr[]);
D. double getAverage(int *arr, int len);
E. double getAverage(int arr[]);
```

4. Which of the following is an example of an INFINITE LOOP?

```
A.  int i=0;
    if(true)
      cout<< i++;
B.  int i=0;
    while(i<5)
      cout<<i++;
C.  for (int i = 0; i < -1 ; i++)
      cout<<i;
D.  for(int i = 4; i > 0; i++)
      cout<<i;
E.  None of the above
```

Name: ______________________

5. Which of the following is a hex representation of the decimal number 62?
   A. 0x1E
   B. 0x2E
   C. 0x3E
   D. 0x101010
   E. None of the above

6. What will be the elements of the integer array `'arr'` after the following code is executed?

```
    int arr[4]={10, 20, 30, 40};
    int i = 2;
    arr[i]= arr[i-1];
    arr[i+1]= arr[i];
```

   A. {10, 20, 30, 40}
   B. {10, 20, 20, 30}
   C. {10, 20, 20, 20}
   D. {10, 30, 40, 40}
   E. None of the above

7. What is the outcome of compiling and executing the following code? Assume it is embedded in a complete and correct C++ program

```
int main(){
      int  x = 0;
      int* p = 0;
      *p = x;
      x = 4;
      cout << *p;
}
```

   A. Compile-time error
   B. Segmentation fault
   C. No error, the output is 0
   D. No error, the output is 4
   E. None of the above

8. Consider the definition of a struct Ball given below. Which of the following C++ statements creates an object of type Ball on the runtime stack, sets the color member variable to “red” and isBouncy to “true”?

```
struct Ball{
    string color;
    bool isBouncy;
};
```

```
A. Ball b;
     color = “red”;
     isBouncy = true;

B. Ball b, *p;
     p = &b;
     p->color = “red”;
     p->isBouncy = true;

C. Ball b;
     b.color = “red”;
     b.isBouncy = true;

D. Options A and B

E. Options B and C
```

9. Which of the following statements correctly creates an integer array with 10 elements on the **heap**?

```
A. int arr = new int;
B. int arr[10];
C. int arr = new int[10];
D. int *arr = new int[10];
E. None of the above
```

10. Which of the following statements is FALSE?

A. A node in a linked-list explicitly stores the memory address of the next node in the list
B. The size of a linked-list must be fixed when it is created and cannot be modified later in a C++ program
C. In C++, the value 0 evaluates to false, everything else evaluates to true
D. The size of the pointer ‘p’ is the same in both these declarations: `char* p;` and `int* p;`
E. None of the above

Name: ______________________

Part 2. Recursion and arrays

1. [5 pts] What is the outcome of compiling and executing the following code. Assume it is embedded in an otherwise correct and complete C++ program

```
  void printBackwards(int *arr, int len){
         if (len <=0) return;
         cout<<arr[len-1]<<" ";
         printBackwards(arr, len);
         arr = arr + 1;
         len = len - 1;
   }

   int main(){
        int arr[4] = {1, 2, 3, 4};
        printBackwards(arr, 4);
   }
```

2. [10 pts] Implement a function that returns the minimum value in an integer array **'arr'** of length **'len'**. You MUST use a RECURSIVE solution.

```
//Precondition: An integer array with length at least one
//Postcondition: Returns the minimum value in the array
int minArrayRecursive(int *arr, int len);
```

Part 3: Linked Lists, Recursion, C-strings

You are given the files: node.h and linkedlist.h. Below is the content of node.h that contains the declarations for **Node** and **LinkedList**. These declarations are similar to those used in the construction of a linked-list in lab06 and lab07, except every node now stores a character instead of an integer. Use these declarations in the questions that follow:

```
//filename: node.h
#ifndef NODE_H
#define NODE_H
struct Node {
  char data;
  Node *next;
};

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

Name: ______________________

Below is the content of linkedlist.h that contains the declaration of functions that you will implement in subsequent questions.

```
//filename: linkedlist.h
#ifndef LINKEDLIST_H
#define LINKEDLIST_H
#include "node.h"
//IMPORTANT: In all the functions the linked-list may only contain
//letters of the alphabet in either lower or upper case

//Precondition: The address of a valid LinkedList and a char value
//that may be an alphabet in either lower or upper case
//Postcondition: Adds a new node with data element set to value to the
//end of the linked list.
void addToEndOfList(LinkedList*& list, char value);

//Precondition: A char array with a given length, not necessarily a
//C-string but containing only letters of the alphabet
//Postcondition: The address of a new LinkedList containing all the
//characters of the input array in the same order, where each node of
//the linked list contains one character of the array.
LinkedList* arrayToLinkedList(char* arr, int len);

//Precondition: The address of a valid LinkedList and a char value
//that is an alphabet in either lower or upper case
//Postcondition: Returns the number of occurrences of the given
//alphabet in either lower or upper case in the linked list
//You must use an iterative implementation (loops).
int countCharIterative(LinkedList* list, char value);

//Precondition: The address of the first node in a linked list and a
//char value that is a letter of the alphabet in either lower or upper
//case.
//Postcondition: Returns the number of occurrences of the given char
// value in the linked list using a recursive implementation.
int countCharHelper(Node* head, char value);

//Precondition: The address of a valid LinkedList and a char value
//that is a letter of the alphabet
//Postcondition: Returns the number of occurrences of the given
//value in the linked list in either upper or lower case. This
//function uses the helper function countCharHelper()
int countChar(LinkedList* list, char value);
#endif
```

Name: ______________________

Assume that a linked-list is already created in memory and is represented by the following pointer diagram: The hex number outside each box is the memory location of the corresponding data object. **`mylist`** is a pointer to a **`LinkedList`** object. **`mylist`** is created on the stack, all other data objects are created on the heap

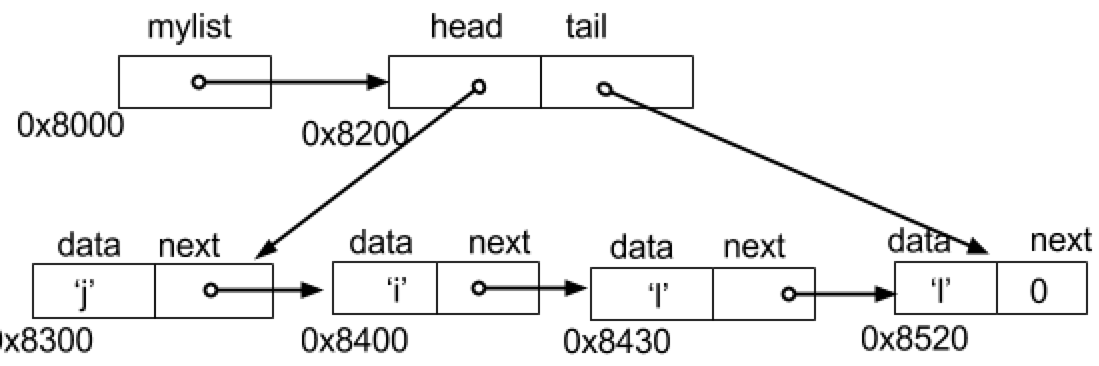

Figure 1: A linked-list containing one alphabet (char) in each node.

1. [5 pts] Given the above diagram, write the output of each of the following C++ statements. Express your answer as an alphabet, a hex number or decimal as appropriate.

```
a. cout<<mylist;
b. cout<<mylist->tail;
c. cout<<mylist->tail->data;
d. cout<<mylist->head->next;
e. cout<<mylist->head->next->data;
```

2. [10 pts] Consider the following function that has an INCORRECT implementation that tries to add a new Node to the end of a linked-list (that may be empty). Answer the questions that follow.

```
void addToEndOfList(LinkedList*& list, char value){
    Node* n = new Node;
    list->tail = n;
    n->data = value;
    n->next = list->tail;
    return;
}
```

a. (3 pts) Without modifying the function, assume that the function is called passing in **`mylist`** (shown above) and the character **`'s'`** as follows: **`addToEndOfList(mylist, 's');`** Starting with the linked list shown above, draw a pointer diagram to show the state of memory right before the function **`addToEndOfList()`** returns. You must label your diagram with the variables: **`list, mylist, value,`** show all the nodes of the linked list and any new objects that are created by **`addToEndOfList()`**.
b. (1 pt) Where is the pointer 'n' created (Stack or Heap)? Fill in the correct circle
c. (1 pt) Does the function **`addToEndOfList()`** have a memory leak? Fill in the correct circle
d. (5 pts) Provide a correct implementation of the function **`addToEndOfList()`** that works on any linked-list including an empty list.

3. [5 pts] Implement the function **`arrayToLinkedList()`** described on page 5. Assume you have a correct implementation of **`addToEndOfList()`** and use it in your implementation of **`arrayToLinkedList()`**. As an example the following code should result in the linked-list depicted in Figure 1 on page 6.

```
char name[]= "Jill";
LinkedList *mylist = arrayToLinkedList(name, 4);
```

In the following questions you may use the standard library function **`char tolower(char c);`** that takes a single character as input and returns the lower-case version of the character.

4. [10 pts] Implement the function **`countCharIterative()`** described on page 5. As an example if **`mylist`** is the pointer to a **`LinkedList`** as depicted in Figure 1 on page 6, then the following calls to **`countCharIterative()`** must return the given values.

```
countCharIterative(mylist, 'k') should return the value 0
countCharIterative(mylist, 'J') should return the value 1
countCharIterative(mylist, 'j') should return the value 1
countCharIterative(mylist, 'l') should return the value 2
```

You have an option to answer either Q5 or Q6. Clearly select one in your answer sheet before attempting it.

5. [10 pts] Implement the function **`countCharHelper()`** described on page 5. Note that your implementation MUST be RECURSIVE. Below is an implementation of **`countChar()`** that uses the **`countCharHelper()`** function and must behave exactly as **`countCharIterative()`**

```
int countChar(LinkedList* list, char value){
     assert(list);
      return countCharHelper(list->head, value);
}
```

OR

6. [10 pts] Implement the following function RECURSIVELY:

```
int countCharInString(char* str, char value)
//Precondition: A valid C-string containing only letters of the
//alphabet in either lower or upper case and a char value that is a
//letter of the alphabet in either lower or upper case
//Postcondition: The number of occurrences of the given alphabet in
//the C-string. The count should be insensitive to case as in all
//the previous questions.
```

Name: ________________________

Scratch Paper

Name: ________________________