CSCI 131-- Techniques of Programming

College of the Holy Cross

Topics for Final Exam:

This sheet is intended to help you prepare for the final exam in this course. The exam will cover all topics in the course, but there will be somewhat more emphasis on the topics covered since exam two. In addition to studying the topics and questions listed here, you should also review the topics and questions presented on the review sheets for exams 1 and 2, since these are all potential subjects for final exam questions. The following lists topics covered since exam 2.

1. Abstract Data Types
    - Definition of abstraction
    - Data Abstraction: Logical Properties vs. Implementation
    - Properties of Abstract Data Type: domain & operations
    - Abstract Data Type specification
    - Different possible implementations
    - Information hiding

2. Classes in C++
    - Class declaration
    - Public and Private members
    - class operations
    - Implementation of class member functions
    - Scope resolution operator
    - Class constructors
    - Object oriented Programming terminology
    - Client code for manipulating class objects
    - Inheritance
3. Searching and Sorting a list

4. Pointers
    - Declaring pointers
    - Allocating memory for a pointer (using "new")
    - The NULL pointer
    - Accessing memory that is pointed to by a pointer
    - Pointer operations
    - Pointers to structs
    - Accessing members of structs using a pointer
    - Diagramming pointers
    - The memory "heap"

5. Linked Lists
    - Creating a list node
    - Adding nodes to a list
    - Removing nodes from a list
    - Accessing list data
    - Using typedef for pointers to lists
    - Traversing a list
6. Recursion
    - Definition of a recursive function
    - Base Case vs. General Case
    - Trace of recursive function
    - Invocation tree for recursive function
    - Tail Recursion
    - Writing Tail recursion as a while loop
    - Using recursion with lists
    - Backtracking

The following problems are intended to help you study for the exam. Note that there will also be questions covering topics from the first two midterms, so you should be sure to review those topics as well. Use your Exams and Review sheets for those exams to review.

1) Write a function that counts how many even numbers there are in an integer linked list. The function should have one parameter, a pointer to a linked list. The function should not alter the list. It should return an integer value equal to the number of even numbers in the list.

You can assume the following definitions:

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

2) Consider the following code:

```
int *y;
int *x;
x = new int;
*x = 15;
*x += 2;
y = x;
*y *= 2;
```

a) Draw a diagram to show represent the pointers as each line of this code is executed.

b) What are the values of *x and *y after executing this code?

3) Consider the following recursive function:

```
int Mystery( char myString[ ], char theChar, int first, int last) {
        int answer = 0;
        if (first > last) {
                return 0;
        } else {
                answer = Mystery ( myString, theChar, first + 1, last);
                if ( myString[first] == theChar) {
                        return answer + 1;
                } else {
                        return answer;
                }
        }
}
```

a) Indicate which lines of code represent the solution to the base case. Also state what the base case is.

b) Indicate which lines of code represent the solution to the general case.

c) Suppose you have the following code use to call the Mystery function:

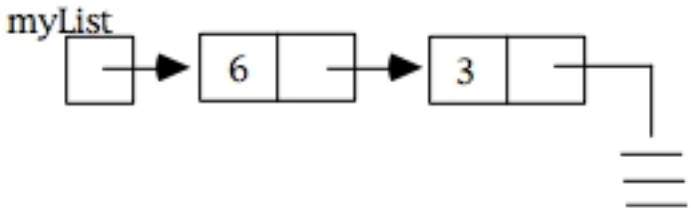
```
char theString = "CSCI";
int test;
test = Mystery(theString, 'C', 0, 4);
```

Draw a trace of the values of the following parameters as each recursive function call is executed:

first	last	myString[first]	answer	return value

d) What does the Mystery function do?

4) Consider the following linked list:



Assume a Node is represented by the following struct:

```
struct Node {
        int  myNumber;
        Node *nextNode;
};
typedef Node *NodePtr;
```

a) Write an expression to refer to the integer in the second node of the list.

b) What is the value of the following expression?

myList - > nextNode - > nextNode

c) Write the C++ code to create and insert a new node, whose data value is 4, in between the two nodes of myList.

5. Consider the following class declarations:

```
class Student {
        public:
                void SetName( char name[ ]);
                void SetID( int idNum);
                void Write( ) const;
                void SetGpa(float the_GPA);
                Student ( );
                Student (char initName[ ], int initId);
        private:
                char name [25];
                int id;
                float gpa;
};

class StudentYear : public Student {
        public:
                void SetYear( int year);
                void Write ( ) const;
                void incrementYear( );
                StudentYear ( );
                StudentYear (char initName[ ], int initId, int initYear);
        private:
                int year;
};
```

a) Which class is the parent (base) class and which is the child (derived) class?

b) Which functions are the same for both the Student and the StudentYear class?

c) What are the private data members for each class?

d) Which private data members can be accessed directly by the member functions of each class?

e) What specific kind of function is StudentYear( )?

f) Write the implementation of the Write( ) function for the Student class. This function should write to the standard output the values of name, id and gpa of the Student object.

g) Write the implementation of the Write( ) function for the StudentYear class. This function should write to the standard output the values of name, id, gpa and year of the StudentYear object.

h) Write the implementation of the Student(char initName[ ], int initId) function for the student class. This function should assign initName[ ] to the name data member and initID to the id data member of the Student object.