**Name:** _____

| | Score | Max |
|---|---|---|
| I. | | 20 |
| II. | | 20 |
| III. | | 20 |
| IV. | | 40 |

**Instructions:**

1. This is a closed-book, closed-notes exam.
2. You have 75 minutes for the exam.
3. Calculators, cell phones and laptops must be put away.
4. Clearly indicate your final answer.

# I. True/False (2 points each)

For each question in this section, indicate whether the statement is TRUE or FALSE. Circle **ONE** answer. Choose the **BEST** answer.

1. The function $17n + 4\log n + 3$ is $O(\log n)$.

   TRUE FALSE

2. The function $17n + 4\log n + 3$ is $O(n)$.

   TRUE FALSE

3. The function $17n + 4\log n + 3$ is $O(n \log n)$.

   TRUE FALSE

4. The function $17n + 4\log n + 3$ is $O(n^2)$.

   TRUE FALSE

5. The function $17n + 4\log n + 3$ is $O(n^3)$.

   TRUE FALSE

6. The function $32n\log n - 94n + 37\log n + 2$ is $O(\log n)$.

   TRUE FALSE

7. The function $32n\log n - 94n + 37\log n + 2$ is $O(n)$.

   TRUE FALSE

8. The function $32n\log n - 94n + 37\log n + 2$ is $O(n \log n)$.

   TRUE FALSE

9. The function $32n\log n - 94n + 37\log n + 2$ is $O(n^2)$.

   TRUE FALSE

10. The function $32n\log n - 94n + 37\log n + 2$ is $O(n^3)$.

    TRUE FALSE

## II. Multiple Choice (4 points each)

For each question in this section, circle **ONE** answer. Choose the **BEST** answer.

1. Assuming enough storage is available, what is the running time to add an item in an unsorted array?

   (a) $O(1)$
   (b) $O(\log n)$
   (c) $O(n)$
   (d) $O(n^2)$

2. What is one disadvantage of storing data in an array?

   (a) Cannot use binary search.
   (b) It takes $O(n)$ time to access an item in the middle of the array.
   (c) If the array is full and we want to add another item, it takes $O(n)$ time to allocate a new array and copy the data.
   (d) all of the above

3. When a linked list is used to implement a stack, what is the running time for the `push` and `pop` operations?

   (a) $O(1)$ to push and $O(n)$ to pop.
   (b) $O(n)$ to push and $O(1)$ to pop.
   (c) $O(n)$ to push and $O(n)$ to pop.
   (d) $O(1)$ to push and $O(1)$ to pop.

4. When an array is used to implement a stack, what is the running time for the `push` and `pop` operations?

   (a) $O(1)$ to push and $O(n)$ to pop.
   (b) $O(n)$ to push and $O(1)$ to pop.
   (c) $O(n)$ to push and $O(n)$ to pop.
   (d) $O(1)$ to push and $O(1)$ to pop.

5. What is one disadvantage of storing data in a linked list?

   (a) Cannot use binary search.
   (b) It takes $O(n)$ time to access the last item of the linked list.
   (c) If the linked list is full and we want to add another item, it takes $O(n)$ time to allocate a new node and copy the data.
   (d) none of the above

## III. Running Times (5 points each)

For each of the following code fragments, estimate an upper bound on the worst case running time of the code fragment using $O()$ notation. Give your answer in terms of $n$.

*Briefly explain your answer.*

1. What is the running time of the following code fragment?

```
t = n ;
while ( t > 1 ) {
   t = t / 2 ;
}
```

Running Time = _____

Explanation:

2. What is the running time of the following code fragment?

```
int something = 0 ;

for (int i=0 ; i<n ; i++) {
   t = i ;
   while ( t > 1 ) {
      t = t / 2 ;
   }
}
```

Running Time = _____

Explanation:

3. What is the running time of the following code fragment?

```
int something = 0 ;
int t = n ;

while (t > 1) {
   for (int i=0 ; i<t ; i++) {
      something++ ;
   }
   t = t / 2 ;
}
```

Running Time = _____

Explanation:

4. What is the running time of the following code fragment?

```
t = 5 * n ;
while ( t < n ) {  // less than!
   t = t + 5 ;
}
```

Running Time = _____

Explanation:

## IV. Coding (20 points each)

The following are class definitions for a singly-linked list of `int` values. You should assume that the list does *not* use a dummy header. Also, note that the data members for `Node` are public.

```
// The node used in List
class Node {
public:
   Node() ;
   Node(int data);
   int m_data;
   Node* m_next;
};


// List is a linked list of ints
class List {
public:
   // Creates a default empty list
   List();

   // Creates a copy of another list
   List(const List &other);

   // Destructor
   ~List();

   // Assignment operator
   const List& operator=(const List &rhs);

   // Insert "data" into the list
   void insert(int data);

   // Remove the last node in the linked list, if it exists
   Node * removeLast() ;

   // Return the number of times that the value n appears
   int count(int n) ;

private:
   Node* m_header ;

};
```

1. Write an implementation of the `count()` member function for `List` as it would appear in a `.cpp` file. The `count()` function should return the number of times that the parameter `n` appears in the linked list. If the list is empty, `count()` should return 0.

2. Write a stand alone function (not a member function) `copyList()` with the following prototype:

```
Node * copyList(Node *ptr) ;
```

This function should assume that `ptr` points to the beginning of a linked list. It should make a "deep copy" of that list. Make sure that you allocate memory for nodes of the new list. The return value is a pointer to the first node of the new list. The intention here is that `copyList()` can be a helper function for the copy constructor and the assignment operator for the `List` class. For example, the copy constructor for the `List` class can be implemented as:

```
List::List(const List& other) {
   m_header = copyList(other.m_header) ;
}
```