Name: ____

Write all of your responses on these exam pages. If you need extra space please use the backs of the pages.

1 Short Answer: 5 Points Each

1. State the precise mathematical definitions of Big-O, Big-Ω, and Big-Θ. Also give the common meaning of each, specifically, what bound does it indicate?

2. Fill out the time complexity table below.

Algorithm	**Best**	**Average**	**Worst**
Bubble Sort			
Insertion Sort			
Selection Sort			
Quick Sort			
Merge Sort			
Linear Search on Array			
Binary Search on Sorted Array			

3. Using the definition of $O(f(n))$, prove that $T(n) = 3n^2 + 5n + 1$ is $O(n^2)$.

4. Write a recursive function that will compute the double factorial. The double factorial is defined as

$$n!! = n \cdot (n-2) \cdot (n-4) \cdots 1$$

and $0!! = 1$. For example, $3!! = 3$, $4!! = 8$, $5!! = 15$, $6!! = 48$, $7!! = 105$, $\ldots$.

5. Write a templated recursive binary search function for an array, assume the array is already sorted.

6. Write the following Linux console commands to do the following.

   (a) The command to get a directory listing of all files (including hidden files) in long format. You may not use any aliases.

   (b) The command to create a new and empty text file named `main.cpp`.

   (c) The command to move back a single directory (that is from the current directory to the parent directory) and the command to move back to the user's home directory no matter where they are.

   v

7. Draw the binary search tree after the following values have been inserted in this order.

   `19, 18, 31, 25, 32, 20, 7, 10, 9`

   Then draw the tree after the root node has been removed.

8. Write a complete make file for compiling the following code project. The compilation must do the minimal amount of recompiling dependent on the changes in the files. Also, the recompilation should track changes in all dependent files. The project to be compiled has three files in it, main.cpp, NumberList.h and NumberList.cpp. The main.cpp and NumberList.cpp both include NumberList.h. There is no templating in these class structures so the specification and implementation of the NumberList class is divided between the two files. Use an arrow "⟶" to denote a tab character.

2 Coding Exercises: 20 Points Each

1. This exercise is to write a linked list class with some basic functionality. The class is to be templated so that it can store any data type that supports assignment, streaming out, and equality testing (i.e. ==). Specifically, the class structure is to implement the following. As usual, there is not to be any inline code in the specification for any of the functions.

   - The list node should be an internal private struct (or class) named `ListNode`.
   - A default constructor only.
   - A destructor, obviously. Make sure that there are no memory leaks.
   - `appendNode` that takes a single parameter, the element to be added to the list, and appends the element on to the end of the list.
   - `insertFront` that takes a single parameter, the element to be added to the list, and puts the element on to the front of the list.
   - `deleteNode` that takes a single parameter, the element to be deleted from the list, and removes the first occurrence of the element from the list. If the element is not in the list then the list is unaltered.
   - `displayList` that will display the list to the console screen horizontally.

   The following three pages are for you answer to this exercise. There is a sample program below and its output. Read this very closly, your class structure is to produce exactly the same output to this sample code. The next three pages are for your answer.

```cpp
#include "LinkedList.h"
#include <iostream>

using namespace std;

int main() {
    LinkedList<int> list;

    list.appendNode(5);
    list.appendNode(2);
    list.appendNode(1);
    list.appendNode(3);
    list.appendNode(7);

    list.displayList();

    list.insertFront(10);
    list.insertFront(15);
    list.insertFront(25);

    list.displayList();

    list.deleteNode(2);
    list.displayList();
    list.deleteNode(15);
    list.displayList();
    list.deleteNode(12345);
    list.displayList();
    list.deleteNode(7);
    list.displayList();
    list.deleteNode(25);
    list.displayList();

    return 0;
}
```

Output

```
5 2 1 3 7
25 15 10 5 2 1 3 7
25 15 10 5 1 3 7
25 10 5 1 3 7
25 10 5 1 3 7
25 10 5 1 3
10 5 1 3
```

LinkedList Class

LinkedList Class

LinkedList Class

2. Write the implementation of either the templated Quick Sort algorithm we did in class or the templated Merge Sort algorithm we did in class. Do only one of these.

3. This exercise is to code portions of the integer binary search tree we discussed in class. The specification for the class is below.

```
class IntBinaryTree {
private:
    struct TreeNode {
        int value;
        TreeNode *left;
        TreeNode *right;
    };

    TreeNode *root;

    void insert(TreeNode*&, TreeNode*&);
    void destroySubTree(TreeNode*);
    void deleteNode(int, TreeNode*&);
    void makeDeletion(TreeNode*&);
    void displayInOrder(TreeNode*) const;
    void displayPreOrder(TreeNode*) const;
    void displayPostOrder(TreeNode*) const;
    void IndentBlock(int);
    void PrintTree(TreeNode*, int, int);

public:
    IntBinaryTree() { root = nullptr; }
    ~IntBinaryTree() { destroySubTree(root); }

    void insertNode(int);
    bool searchNode(int);
    void remove(int);

    void displayInOrder() const { displayInOrder(root); }
    void displayPreOrder() const { displayPreOrder(root); }
    void displayPostOrder() const { displayPostOrder(root); }
    void PrintTree(int Indent = 4, int Level = 0);
};
```

Code only the following functions. Your implementation should be written as functions that are outside the specification. No inline code.

- Write the `insertNode` and the `insert` functions that will collectively insert a node into the tree in the correct position.
- Write the `searchNode` function that will return true if the value being searched for is in the tree and false otherwise.
- Write the `remove`, `deleteNode`, and the `makeDeletion` functions that will collectively delete a node from the tree.

Your code for the `insertNode` and the `insert` functions.

Your code for the `searchNode` function.

Your code for the `remove`**,** `deleteNode`**, and the** `makeDeletion` **functions.**

3 Extra Credit: 5 Points

1. Prove that $T(n) = 2^n$ is $O(n!)$ and $T(n) = n!$ is not $O(2^n)$.