

Name: \_\_\_\_\_

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

1. (*15 points*) Pointers & Dynamic Arrays: Write a function that takes as parameters the pointers of two sorted integer arrays and the two array sizes, and returns the pointer of a new integer array that is the sorted merging of the two arrays. For example, if the two parameter arrays are [3, 7, 10, 16, 22] and [1, 2, 5, 10, 12, 17] the resulting array is [1, 2, 3, 5, 7, 10, 10, 12, 16, 17, 22].

2. (15 points) Basic Classes: Write the specification and implementation (no inline code) for a Rectangle class that stores a height and width (decimal values possible), constructor that takes the height and width as well as a default constructor that sets height and width to 1 each. The non-default constructor should error check the input so that any negative input is changed to 0. Accessors and mutators, mutators should do the same error checking. A member function for the area and a member function for the perimeter.

3. (20 points) Inheritance & Polymorphism: In this exercise we construct class structures, automobile, car, and truck. automobile is to be the base class and the other two will be derived classes off of this. For the mutators listed below you do not need to do any error checking on the values.

- automobile is to store the name of the auto that can be set by its constructor but defaults to “auto”. It also stores the mileage of the vehicle. It also has a destructor, member functions to access and change the name and mileage, and a virtual function toString that will print out the name and mileage.
- car inherits off of automobile, it further stores the number of doors the car has. It has a constructor allowing input of a name and a default that uses “car” as the name. It also has a destructor and accessor and mutator for the doors. Its toString function will, in addition to printing out the name and mileage, will print out the number of doors.
- truck inherits off of automobile, it further stores the length of the truck bed. It has a constructor allowing input of a name and a default that uses “truck” as the name. It also has a destructor and accessor and mutator for the bed length. Its toString function will, in addition to printing out the name and mileage, will print out the length of the bed.

With these classes and functions in place the following program will produce the output below.

```
#include <iostream>
#include <vector>

#include "auto.h"
#include "car.h"
#include "truck.h"

using namespace std;

int main() {
    vector<automobile *> autos;
    car *carptr = new car;
    carptr->setDoors(4);
    carptr->setMiles(80000);
    autos.push_back(carptr);

    truck *truckptr = new truck("Sport Pickup");
    truckptr->setBedLength(6);
    truckptr->setMiles(35000);
    autos.push_back(truckptr);

    truckptr = new truck;
    truckptr->setBedLength(8);
    truckptr->setMiles(125300);
    autos.push_back(truckptr);

    carptr = new car("Corvette");
    carptr->setDoors(2);
    carptr->setMiles(15200);
    autos.push_back(carptr);

    for (auto v : autos)
        cout << v->toString() << endl;

    return 0;
}
```

Output:

```
car Miles: 80000 Doors: 4
Sport Pickup Miles: 35000 Bed Length: 6
truck Miles: 125300 Bed Length: 8
Corvette Miles: 15200 Doors: 2
```

**automobile class code**

---

**car class code**

---

**truck class code**

---

4. (20 points) Operator Overloading: Say we have a class structure called `Point` that represents a point  $(x, y, z)$  in three dimensions. The data,  $x$ ,  $y$ , and  $z$  are stored in an array of three doubles, specifically, `double P[3];` is its declaration. Write the following overloads for the  $+$ ,  $-$  and  $*$  operators. Give both the specification and implementation of each with no inline code. You may assume that the `Point` class has a non-default constructor that takes three double parameters, specifically, `Point p(x, y, z);` is a valid creation of the point `p`.
- The addition of two points is another point and is done by adding the respective  $x$ ,  $y$ , and  $z$  values. So in mathematical notation,  $(x_1, y_1, z_1) + (x_2, y_2, z_2) = (x_1 + x_2, y_1 + y_2, z_1 + z_2)$ .
  - The subtraction of two points is another point and is done by subtracting the respective  $x$ ,  $y$ , and  $z$  values. So in mathematical notation,  $(x_1, y_1, z_1) - (x_2, y_2, z_2) = (x_1 - x_2, y_1 - y_2, z_1 - z_2)$ .
  - Multiplying a number and a point (called scalar multiplication) is just multiplying each entry by that number. So in mathematical notation,  $a \cdot (x_1, y_1, z_1) = (a \cdot x_1, a \cdot y_1, a \cdot z_1)$ . This can also be done on either side of the point, that is,  $(x_1, y_1, z_1) \cdot a = (a \cdot x_1, a \cdot y_1, a \cdot z_1)$ .

5. (30 points) Linked Lists:

- (a) Write an `appendNode` function for the templated linked list. The function will take in a single parameter of the templated type, create the node of `ListNode` type and attach it to the end of the list. The start of the list is a pointer called `head`. This implementation of the linked list does not have a tail pointer.



- (b) Write a copy constructor for the templated linked list class. Assume the class name is `LinkedList`.

- (c) Write a `deleteNode` function for the templated linked list. The function will take in a single parameter of the templated type and remove the first occurrence of that value from the list. If the value is not in the list then the list is unaltered. The start of the list is a pointer called `head`. This implementation of the linked list does not have a tail pointer.

6. (10 points) Stacks, Queues, and STL: Write a function called `balanced` that takes in a string (which represents a mathematical expression) and uses an STL stack to determine if the parentheses are balanced. The function will return `true` if the parentheses are balanced and `false` otherwise. For example, an input of `(x+4/(x-1))* (y+7)/(z*(x + 4*y))` would return `true` and an input of `x+4/(x-1))` would return `false`.

7. (20 points) Recursion:

- (a) The  $D$  sequence is defined to be  $D(1) = 1$ ,  $D(2) = 1$ , and

$$D(n) = D(D(n-1)) + D(n-1 - D(n-2))$$

Write a recursive function  $D$  that takes in a single long parameter  $n$  and returns a long that is the value of  $D(n)$ .

- (b) Write a recursive function that calculates  $n$  choose  $k$ ,

$$\binom{n}{k}$$

The mathematical recursive definition is,

$$\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1}$$

If  $n$  or  $k$  is 0 then the value is 1, if  $n = k$  the value is 1, and we will define anything outside these ranges the value is 0. That is, if  $n$  or  $k$  are negative or if  $k > n$  we will simply have the program return 0.

- (c) In a standard templated linked list class with external templated `ListNode` class we will rewrite the display list function to be recursive. The private (or protected) section contains the specification,
- ```
void displayListRec(ListNode<T> *) const;
```
- The public section contains the specification,
- ```
void displayList() const { displayListRec(head); }
```
- Write the `displayListRec` function recursively, that is, no loops allowed.

## 8. (25 points) Binary Search Trees:

- (a) Write the recursive insert function for the templated binary search tree class. This function is to take a pointer to a tree node and a pointer to a new node and insert the new node in the correct position in the BST. That is, if we were to run the following code, the insert function will put this new node in the correct position in the tree.

```
TreeNode *newNode = new TreeNode;
newNode->value = item;
newNode->left = newNode->right = nullptr;
insert(root, newNode);
```

- (b) Write an in-order traversal function for the templated binary search tree class that displays the node value to the console screen.

- (c) Write the three templated functions,

```
void deleteNode(T, TreeNode *&);  
void makeDeletion(TreeNode *&);  
void remove(T);
```

That together will delete an item from a binary search tree.





9. (20 points) Complexity:

- (a) State the precise mathematical definitions of Big- $O$ , Big- $\Omega$ , and Big- $\Theta$ . Also give the common meaning of each, specifically, what bound does it indicate?

(b) Fill out the time complexity table below.

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

(c) Prove that  $T(n) = 2^{\sqrt{\lg n}}$  is  $O(\sqrt[5]{n})$ .

10. (15 points) Sets and Multisets: Write a program that will ask the user to input the number of dice to roll, the number of sides of the dice (assume all die are the same), and the number of rolls. Have the program roll all the dice that many times and store each result in a multi set. After all the data is stored, have the program print out the counts of each roll. Make sure that each roll total appears only one time. Example runs are below.

```
Input number of dice to roll: 2
Input number of sides for each die: 6
Input number of rolls of dice: 10
Results
4 1
6 3
7 3
8 1
9 1
10 1
```

```
Input number of dice to roll: 2
Input number of sides for each die: 4
Input number of rolls of dice: 1000000
Results
2 61954
3 124703
4 187578
5 250120
6 187825
7 125267
8 62553
```

11. (*10 points*) Sorts: Write the templated Quick Sort functions to sort an array.

12. (10 points) Short Answer:

- (a) Assuming that `ptr` is a pointer to a double, what memory address is `(ptr + 10)`?
- (b) Describe the difference between an instance member variable and a static member variable.
- (c) In what circumstances is the copy constructor called?
- (d) What is the difference between a protected class member and a private class member?
- (e) Which destructor is called first, that of the derived class or the base class? What must be done to ensure that the entire chain of destructors in an inheritance chain are called?
- (f) What is an iterator? Write the definition of one for a vector of longs.

- (g) 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 five files in it, `main.cpp`, `IntBST.h`, `IntBST.cpp`, `NumberList.h` and `NumberList.cpp`. The `main.cpp` and `NumberList.cpp` both include `NumberList.h` and the `main.cpp` and `IntBST.cpp` both include `IntBST.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.