Data Structures - Final Exam

Name:_____

Question 1 (60 points)

1.1 Consider the function `f()` on the list object `alist`.

```
void f(list<int>& alist)
{
  list<int>::iterator iter ;
  int j;

  while(alist.size() > 1)
    {
      iter = alist.begin();
      for(j = 0 ;  j < alist.size()-1; j++)
         iter++;
      cout << *iter << "  ";
      alist.erase(iter);
    }
}
```

Assume list `alist` has values `7 12 9 8 15`. What is the output from `f()`?

1.2 Consider the following function `f`.

```
template <typename T>
void f(vector<T>& v) {
  int i, n;
  n = v.size();
  for (i = 2; i < n; i++){
    v[i-2] = v[i];
  }
  v.resize(n-2);
}
```

Assume the vector `v` has values <`1, 2, 3, 4, 5`>. What is the content of `v` after the function call `f(v)`?

1.3 Describe the behavior of the function `f` defined below.

```
template <typename T>
void f(T arr[], int n){
  queue<T> q;
  int i;

  for (i = 0; i < n; i++)
```

```
    q.push(arr[i]);

  i = 0;
  while (!q.empty()){
    arr[i] = q.front();
    q.pop();
    i++;
  }
}
```

Assume `arr` contains values <1, 5, 4, 3, 2>. What is the content of `arr` after the function call `f(arr,5)`?

1.4 The function `createIntArray(n)` creates a dynamic array of integers whose elements are $< 1, 2, \ldots, n >$ and returns a pointer to the array as the return value. Complete the function definition.

```
int *createIntArray(int n){

}
```

1.5 Consider the function

```
template <typename T>
void f(dnode<T> * & header)
{
  dnode<T> *p = header->next, *q, *r;

  while (p != header)
    {
      q = p->next;
      while (q != header)
        if (p->nodeValue == q->nodeValue)
          {
            r = q;
            q = q->next;
            r->prev->next = q;
            q->prev = r->prev;
            delete r;
          }
        else
          q = q->next;
      p = p->next;
    }
}
```

Assume a doubly linked list contains nodes that have the characters in "grammar". Call f() on the list and then copy the node values back to a string. The resulting string is

---

1.6 Consider a tree created by the following statement (refer to Question 3 for the definition of `tnode`):

```
root = new tnode<char> ('*',
                        new tnode<char>('+',
                                        new tnode<char>('3'),
                                        new tnode<char>('6')),
                        new tnode<char>('-',
                                        new tnode<char>('5'),
                                        new tnode<char>('7')),
                        );
```

1. Display the tree.
2. Give the postorder traversal of the tree.
3. Give the inorder traversal of the tree.

1.7 Show the output of the following program:

```
int main(){
  int i;
  int arr[] = {6, 3, 3, 4, 9, 6, 9, 3};
  set<int> s(arr, arr+sizeof(arr)/sizeof(int));
  set<int>::iterator setIter;

  s.insert(10);
  s.insert(2);
  setIter = s.begin();
  while (setIter != s.end()){
    cout << *setIter << " ";
    setIter++;
  }
  cout << endl;
}
```

1.8 The member function `find(x)` in the `map` class searches the map for an element with x as key and returns an iterator to it if found, otherwise it returns an iterator to `map::end`. Show the output of the following program:

```
void output_map(const map<int,string>& m, int from, int to) {
  map<int,string>::const_iterator mapIter;

  for (int i=from;i<=to;i++){
    mapIter = m.find(i);
    cout << i << "   " << (*mapIter).second << endl;
  }
}
```

```
int main(){
  map<int,string> m;

  m[0] = "zero";
  m[1] = "ichi";
  m[2] = "ni";
  m[3] = "san";

  output_map(m,1,2);
}
```

Question 2 (20 points)

Assume that the roster of CISC 3130 is represented as a list of `student` objects where the class `student` is defined as follows:

```
class student {
public:
  string name;
  string id;
  float grade;

  student(string &n, string &i, float g) :
    name(n), id(i), grade(g)
  {}
};
```

The variable `grade` is assumed to hold a real number between 0.0 and 4.0. Write the following functions:

(a) `bool find(const list<student>& roster, const float g)`
This function returns true if there is at least one student in the roster whose grade is greater than or equal to `g`.

(b) `void insert_sorted(list<student>& roster, student &stu)`
Assume that `roster` is sorted by grade from the highest grade to the lowest grade, this function inserts a student `stu` into `roster` such that the roster remains sorted after the insertion.

(c) `pair<float,float> low_high(const list<student>& roster)`
Assume that `roster` is sorted by grade from the highest grade to the lowest grade, this function returns a pair whose `first` member is the lowest grade and `second` member is the highest grade of the roster.

(d) How would you write the function `low_high` in (c) if the roster is not sorted?

Question 3 (20 points)

Given the following `tnode` class template:

```
<template <typename T>
class tnode {
public:
    T data;
    tnode<T> *left, *right;
};
```

write the following functions:

(a)

```
template <typename T>
int occurs(tnode<T> *root, const T &elem);
```

This function returns the number of times `elem` appears in the tree whose root is `root`. Note that the tree may not be a binary search tree.

(b)

```
template <typename T>
bool equals(tnode<T> *root1, tnode<T> *root2);
```

This function returns true if the two binary trees have the same shape and value, i.e., if every corresponding pair of nodes in the two trees have the same value and the same number of children.

(c)

```
template <typename T>
bool search_tree(tnode<T> *root);
```

This function returns true if the binary tree with the given root is a binary search tree.