Question 1: Do questions 1.1-1.5 (3 points each)

1.1 Consider the following function \( f \).

```cpp
template <typename T>
void f(vector<T>& v) {
    int i, j;
    i = 0; j = v.size() - 1;
    while (i < j) {
        T tmp;
        tmp = v[i]; v[i] = v[j]; v[j] = tmp;
        i++; j--;
    }
}
```

Assume the vector \( v \) has values \( <10, 20, 30, 40, 50> \). What is the content of \( v \) after the function call \( f(v) \)?

1.2 Show the output of the following program:

```cpp
int main() {
    int i;
    int arr[] = {6, 3, 3, 9, 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;
}
```
1.3 Consider the class `tnode` defined below:

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

tnode()
{
}

tnode (const T& item, tnode<T> *lptr = NULL, tnode<T> *rptr = NULL):
    nodeValue(item), left(lptr), right(rptr)
{
}
};
```

Display the tree root created by the following statements.

```cpp

d = new tnode<int> (6);
c = new tnode<int> (9,NULL,d);
b = new tnode<int> (4,c,NULL);
a = new tnode<int> (12);
root = new tnode<int> (10,a, b);
```

1.4 Describe the action of `ct`, and give the output of the function call `ct(root,5)` where root is the tree created in 1.3.

```cpp
template <typename T>
int ct(tnode<T> *t, Telm){
    if (t == NULL)
        return 0;
    else
        return ct(t->left,elm)+ct(t->right,elm)+(t->nodeValue > elm ? 1 : 0);
}
```
1.5 A queue is used to reorder an array of integers. Describe the ordering in the integer array `arr` after executing function `f()`?

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

    for (i = 0; i < n; i++)
        q.push_back(arr[i]);

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

(a) creates an ordered array in ascending order.
(b) creates an ordered array in descending order.
(c) reverses the order of the elements.
(d) maintains the same order.

**Question 2 (5 points)**

Implement a class named `Organization` which contains a collection of employees and the following three methods:

(a) `Employee& getEmployee(const string& first, const string& last)`
    Return the employee with the given first name and last name.

(b) `Employee& getEmployee(const string& ssn)`
    Return the employee with the given social security number.

(c) `Employee& highestSalary()`
    Return the employee with the highest salary.

The `Employee` class contains the following member variables:

```cpp
class Employee {
    public:
        string firstName;
        string lastName;
        string ssn;
        double salary;
    Employee(string &first, string &last, string &ss, double sa) :
        firstName(first), lastName(last), ssn(ss), salary(sa)
        {};
};
```

It's assumed that no two people have the same name and only one person has the highest salary.
Answer sheet for Question 2
Question 3 (10 points)

Write the following functions.

(a) `occurs(alist, x)` return the number of occurrences of the value `x` in the list `alist`.

```cpp
template <typename T>
int occurs(const list<T>& alist, const T& x);
```

(b) `mode(alist)` return the mode, i.e., the most frequently occurring element in the list `alist`. If two or more elements occur with the same frequency, then return any one. If the list is empty, then throw the `underflowError` exception.

```cpp
template <typename T>
T& mode(list<T>& alist);
```
Question 4 (10 points)

Write the following functions on \texttt{tnode} (see 1.3 for the implementation of \texttt{tnode}).

(a) \texttt{count\_leaves(root)} return the number of leaves in the binary tree \texttt{root}.

\begin{verbatim}
    template <typename T>
    int count_leaves(tnode<T> *root);
\end{verbatim}

(b) \texttt{level(root,elm)} return the level of the value \texttt{elm} in the binary tree \texttt{root} if \texttt{elm} occurs in the tree and -1 otherwise. It\’s assumed that no value occurs more than once in the tree but the tree may not be a binary search tree.

\begin{verbatim}
    template <typename T>
    int level(tnode<T> *root, T &elm);
\end{verbatim}