Submission detail:

A) Submit cpp file for each question with given main function.

B) Please do not copy from each other, if 0 will ge given if more than 90% of code are same between two students.

C) You can submit only the function or the entire program depends on your preference.

Q1: Write function for the template based LinkList. (20pt)

1. Write append() function for LinkList
2. Write remove() function for LinkList
3. Write printList() function for LinkLis.

Use following template, you must only implement the function declared in this class, 0 will be given if you use any helper function or extra function.

```
template <typename T>
class Node {
public:
    T data;
    Node<T> *next;
    Node(const T &d): data(d), next(nullptr) {}
};

template <typename T>
class LList {
private:
    Node<T> *first;
    Node<T> *last;
    int size;
public:
    LList(): first(nullptr), last(nullptr), size(0) {}
    bool empty(){return first == nullptr;}
    int getSize() const{return size;}
    void append(T n);
    void remove();
    void printList();
};
```

Use following main() to test your function.

```
int main() {
```

```
    LList<int> list;
    list.append(700);
    list.append(900);
    list.append(1100);
    list.append(1200);
    list.printList();
    list.remove();
    list.remove();
    list.printList();
}
```

Output from main:

```
700 900 1100 1200
700 900
```

Answer:

Q2: Write function for the template based LinkList. (20pt)

1. Overload [] operator for LinkList
2. Overload output stream operator for LinkList

Use following template, you must only implement the function declared in this class, 0 will be given if you use any helper function or extra function.

```
class Node {
public:
    T data;
    Node<T> *next;
    Node(const T &d): data(d), next(nullptr) {}
};

template <class T>
class LList {
private:
    Node<T> *first;
    Node<T> *last;
    int size;
public:
    LList(): first(nullptr), last(nullptr), size(0) {}
    bool empty(){return first == nullptr;}
    int getSize() const{return size;}
    void append(T n);
    void remove();
    void printList();
    T& operator[](int index);
    friend ostream& operator<<(ostream& os,LList<T> &l);
```

```
};
```
Use following main() to test your function.

```
int main() {
   LList<int> list;
   list.append(700);
   list.append(900);
   list.append(1100);
   list.append(1200);
   cout<<list;              // 700 900 1100 1200
   cout<<list[0]<<endl;  // 700
   cout<<list[1]<<endl;  // 900
   cout<<list[2]<<endl;  // 1100
   cout<<list[3]<<endl;  // 1200
   list[1] = 3000;
   list[3] = 5000;
   cout<<list;              // 700 3000 1100 500
}
```

Output from main:

```
700 900 1100 1200
700
900
1100
1200
700 3000 1100 5000
```

Answer:

Q3: Sorting (20pt)

If an array of Integer contains the following elements, what would the array look like after the every pass of bubble sort, insertion sort, selectionSort, Merge sort and Quick sort sorting from low to high?

```
99 44 55 2 24 9
```

Answer:
Bubble sort:

Insertion sort:

Selection sort:

Merge sort:

Quick sort:

Q4 (20pt): Write a function that takes an array of vector, and return a set contain shared element between each vector in the vector array.

```
template <class T>
set<T> common(vector<int> v[],int size){
        … …
}
```

Use following main() to test your function.

```
int main(){
  vector<int> v[4];
  v[0] = vector<int>{1,2,3,4,7};
  v[1] = vector<int>{1,4,6,7};
  v[2] = vector<int>{4,7,8,10};
  v[3] = vector<int>{1,4,7,10,11,12};
  set<int> s = common(v,4);
  for(int i: s) cout<<i<<" ";
}
```

Output from main:

```
4  7
```

Q5 (20pt): Write a function call second which takes a vector, and return the second most appeared element from the vector, and return these element in a set. It should work for any data type. (Include all in the set if there are multiple elements are rank second)

```
template <class T>
set<T> second(vector<T> v){
        … …
}
```

Example 1:

Input to function: vector<char> {'h','e','l','l','o','w','o','r','l','d'}
Return from function: set<char> {'o'}

Example 2:

Input to function: vector<int> {1,2,3,1,2,3}
Return from function: set<int> {}

Example 3:

Input to function: vector<double> {1.0,1.0,2.0,2.0,2.0,3.0,3.0}
Return from function: set<double> {1.0,3.0}

I will use extra case to check your function, so please only provide function.

Answer: