Practical 4

Question 1

a)//Yeu Yang

No, we will not be able to enter more entry into the list if the list has reached the maximum capacity.

b)//Yin Lam

Expand the capacity of the array

```
private void expandArray(int newLength){
      //create a new array
      T[] newArray = (T[]) new Object[newLength];

      //copy all entries from the old to the new
      for(int i = 0; i < array.length; i++){
                newArray[i] = array[i];
        }

      //let the array pointer pointing to the new array
      array = newArray;
}
```

//Yong Chen

isArrayFull()

```
private boolean isArrayFull() {
    return length == array.length;
}
```

//Modify the add methods

//Yong Kang

```
public boolean add(T newEntry) {

      boolean isSuccessful=false;

      if(isArrayFull())
      expandArray(length);

      array[length] = newEntry;
      length++;
      isSuccessful= true;


      return isSuccessful;
}
```

```
//Yong Kit
public boolean add(int newPosition, T newEntry) {
    boolean isSuccessful = true;

    if ((newPosition >= 1) && (newPosition <= length + 1)) {
      if (isArrayFull()) {
        expandArray(length);
        makeRoom(newPosition);
        array[newPosition - 1] = newEntry;
        length++;
      }
      else{
        makeRoom(newPosition);
        array[newPosition - 1] = newEntry;
        length++;
      }

    } else {
      isSuccessful = false;
    }

    return isSuccessful;
}
```

Question 2

(a)

A set is a collection of distinct objects. Typical operations on a set include:

//Lee Yong

boolean add(T newElement)

● add: Add a new element to the set

Parameter: newElement

Precondition: Check whether there is a duplicate element

Postcondition: The element added with newElement

Return: Return a true if the element added successfully

//Choon Peng

boolean remove(T anElement)

● remove: Remove an element from the set

Parameter:an element

Precondition:Set have the specific element.

Postcondition:The value of element will be remove from the set

Return:return true if removal was successful or false otherwise

//Dih Yong

boolean checkSubset(Set anotherSet)

● checkSubset: Check if another set is a subset of the current set

Description:To check if another set is a subset of the current set

Parameter:anotherSet

Precondition:check the element in the currentSet and anotherSet
Postcondition:check whether the anotherSet is subset or not
Return:return true if anotherSet is subset of the current set else return false

```
a = {1,2,3,4}
b = {1,2,3,4,5}

a.anotherSet(b) false 5 is not in the set of a
```

//Han Yao
void union(Set anotherSet)
● union: Add another set to the current set
Parameter : anotherSet
Precondition : Check whether the element in the anotherSet is duplicated in currentSet element
Postcondition: anotherSet elements will append in currentSet

//Hao Han
Set intersection(Set anotherSet)
● intersection: Returns a set with elements that are common in both the current set and the given set.

Description: To return a set with elements that are common in both the current set and the given set.
Parameter: another set
Precondition: check whether the common elements in the current set and given set.
Postcondition: common elements in the current set and given set.
Return: return intersection elements

//Joan
boolean isEmpty()
● isEmpty: Check to see if the set is empty
Description: To check whether the set is empty
Precondition: None
Postcondition: None
Return: return true if number of elements equals zero else false

(b) Implement the Set ADT. Use array-based implementation. You may include any necessary utility methods.

```
public interfaces SetInterface<T>{
	boolean add(T newElement);
	boolean remove(T anElement);
	boolean checkSubset(SetInterface anotherSet);
	void union(SetInterface anotherSet);
	SetInterface intersection(SetInterface anotherSet);
	boolean isEmpty();
}
```

```
public class ArraySet<T> implements SetInterface<T>{
	//Hui Shuang - declaration of the data field;
	private T[ ]  array;
	private int length;
	private static final int DEFAULT_CAPACITY = 5;

	public ArraySet(){
		this(DEFAULT_CAPACITY);
	}

	public ArraySet(int initialCapacity){
		length = 0;
		array = (T[ ]) new Object [initialCapacity];
	}


	//JiaJian
	public boolean hasElement(T element){
		for(int i=0;i<length;i++){
			if(array[i].equals(element))
				return true;
		}
		return false;
	}

	@Override
	boolean add(T newElement){

		array[length] = newElement;
		length++;
		return true;
	}









	//Jun Yan
	void removeGap(int givenPosition) {
		for (int index = givenPosition; index < length; index++)
			array[index-1] = array[index];
	}
```

```
A = {1,2,3,4}

A.remove(5);

found=-1

public int findIndex(T anElement) {
      int found = -1;
    for (int index = 0;index < length; index++) {
      if (anElement.equals(array[index])) {
        found = index;
      }
    }

    return found;
}

@Override
boolean remove(T anElement){

      int givenPosition = findIndex(anElement);

      if ((givenPosition >= 1) && (givenPosition <= length)) {
         //result = array[givenPosition];

            if (givenPosition < length) {
                 removeGap(givenPosition);
            }
            length--;
          return true;
      }
      else
            return false;
}
```

```
//Kah Yee
@Override
boolean checkSubset( Set anotherSet){

	if(anotherSet instanceOf ArraySet){
			ArraySet arraySet = (ArraySet) anotherSet;

		T[] anotherSetElement = (T[])arraySet.getArray();
		//getter

		if(arraySet.length > length){
			return false;

		}else{

			for(int i = 0; i < anotherSet.length; i++){
				if(hasElement(arraySet[i] != true){
					return false;
				}
			}

			return true;

		}

	}
	return true;
}

//Kean Min
public void union(SetInterface anotherSet){
//check anotherSet is an instance of ArraySet then cast it to
//ArraySet
	int anotherSetLength = anotherSet.getLength();
	T[] result = (T[]) new Object[length + anotherSetLength];
	//Consider using the add() method
	System.arraycopy(setElement, 0, result, 0, length);
	System.arraycopy(anotherSet.getSetElement(), 0, result,
	length, anotherSetLength);

	length += anotherSet.getSetElement().length;
	setElement = result;
}


//Kuan Xian
SetInterface intersection(SetInterface anotherSet) {
	SetInterface<T> resultSet = new ArraySet<T>();
```

```
            if(anotherSet instanceof ArraySet){
                ArraySet aSet = (ArraySet) anotherSet;

                for(int i = 0; i < aSet.numberOfElements; i++){
                    boolean found = false;

                    for(int j = 0; j < numberOfElements && !found; j++){
                        if(aSet.setArray[i].equals(setArray[j])){
                            found = true;
                        }
                    }

                    if(found)
                        resultSet.add((T) aSet.setArray[i]);
                }
            }

            return resultSet;

        }


    //Lee Ling
    public boolean isEmpty(){
          if(length == 0)
                return true;
          else
                return false;
    }
}
```

(c) Write a driver program to test your implementation of the Set ADT.

(d) Modify the Java interface and class from part b such that your implementation provides an iterator to the set object. Then, include a method in your driver program that uses the iterator to display all the elements in the set object.