



King Fahd University of Petroleum & Minerals
College of Computer Science and Engineering
Information and Computer Science Department
Second Semester 102 (2010/2011)

ICS 201 - Introduction to Computing II

Final Exam
Saturday June 11th 2011
Time: 120 minutes

Name:

ID#:

Please circle your section number below:

Section	01	02	03	04
Instructor	Irfan	Tarek	Sami	Sukari
Day and Time	SMW 10-10:50	SMW 8-8:50	SMW 9-9:50	SMW 13:00-13:50

Question #	Maximum	Obtained
1	20	
2	15	
3	15	
4	25	
5	25	
Total	100	

Question 1: (Output) [20 points]

a) What are the expected outcomes of this program? (5 Points)

```
import java.util.*;

class IteratorDemo {

public static void main(String args[]) {

ArrayList al = new ArrayList();

al.add("C");

al.add("A");

al.add("E");

al.add("B");

al.add("D");

al.add("F");

System.out.print("Original contents of al: ");

Iterator itr = al.iterator();

while(itr.hasNext()) {

    Object element = itr.next();

    System.out.print(element + " ");

}

System.out.println();

ListIterator litr = al.listIterator();

while(litr.hasNext()) {

    Object element = litr.next();

    litr.set(element + "+");

}
```

```
}  
System.out.print("Modified contents of al: ");  
itr = al.iterator();  
while(itr.hasNext()) {  
    Object element = itr.next();  
    System.out.print(element + " ");  
}  
System.out.println();  
System.out.print("Modified list backwards: ");  
while(litr.hasPrevious()) {  
    Object element = litr.previous();  
    System.out.print(element + " ");  
}  
System.out.println();  
} }
```



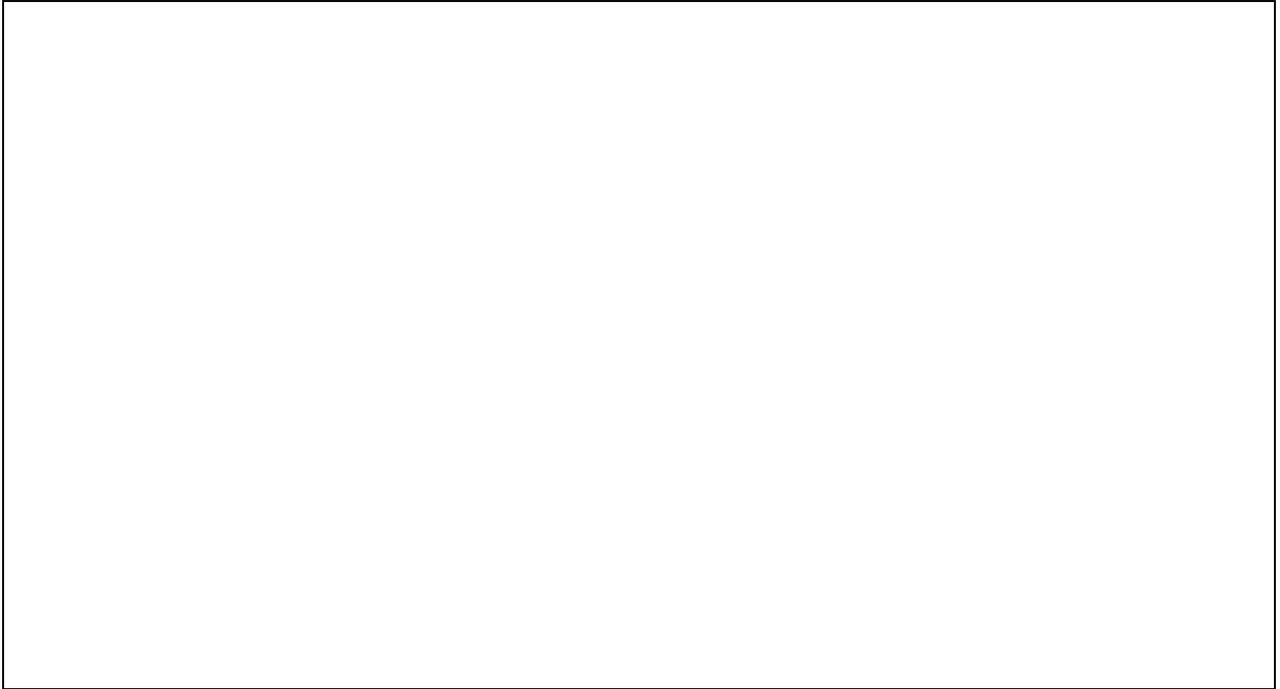
b) What does this loop do? Is it error free code? If no, try to fix it. (5 Points)

```
ArrayList<String> alist = new ArrayList<String>();  
// . . . Add Strings to alist  
int i = 0;  
for (Iterator<String> it = alist.iterator(); it.hasNext(); ) {  
System.out.println(alist.get(i++));  
}
```

c) What are the expected outcomes of this program? (10 Points)

```
import java.util.*;  
class LinkedListDemo{  
public static void main(String args[]){  
LinkedList l1=new LinkedList();  
System.out.println("Original Contents : " + l1);  
l1.add("C");  
l1.add("E");  
l1.add("B");  
l1.add("D");  
l1.add("F");  
l1.addLast("Z");  
l1.addFirst("A");  
l1.add(1,"A2");  
System.out.println("Original Contents after add : " + l1);  
l1.remove("F");  
l1.remove(2);  
System.out.println("Original Contents after removal : " + l1);  
l1.removeFirst();  
l1.removeLast();
```

```
System.out.println("Original Contents : " + l1);  
Object val=l1.get(2);  
l1.set(2,(String) val + "Changed");  
System.out.println("l1 after change : " + l1);  
}  
}
```



Question 2 (Recursion) [15 points]

- a) Write a recursive method `countZeros` which counts the number of zeros in an array of integers. The signature of the method must be:

```
public int countZeros(int[] a)
{
}
}
```

- b) What is the problem of the above method?

- c) Modify your implementation in a) to fix the problem in b) where you can choose the signature you like for the method `countZeros`. That is, you can add parameters, etc.

Question 3 (Generics and Collections) [15 points]

a) What are generics in Java? Give three examples of Generics.

b) Can any type be a parameter for a Generic? Justify your answer.

c) Inside a Generic class, the type parameter cannot be used as an ordinary type. Give an example of an instruction where a normal type can be used but a type parameter cannot.

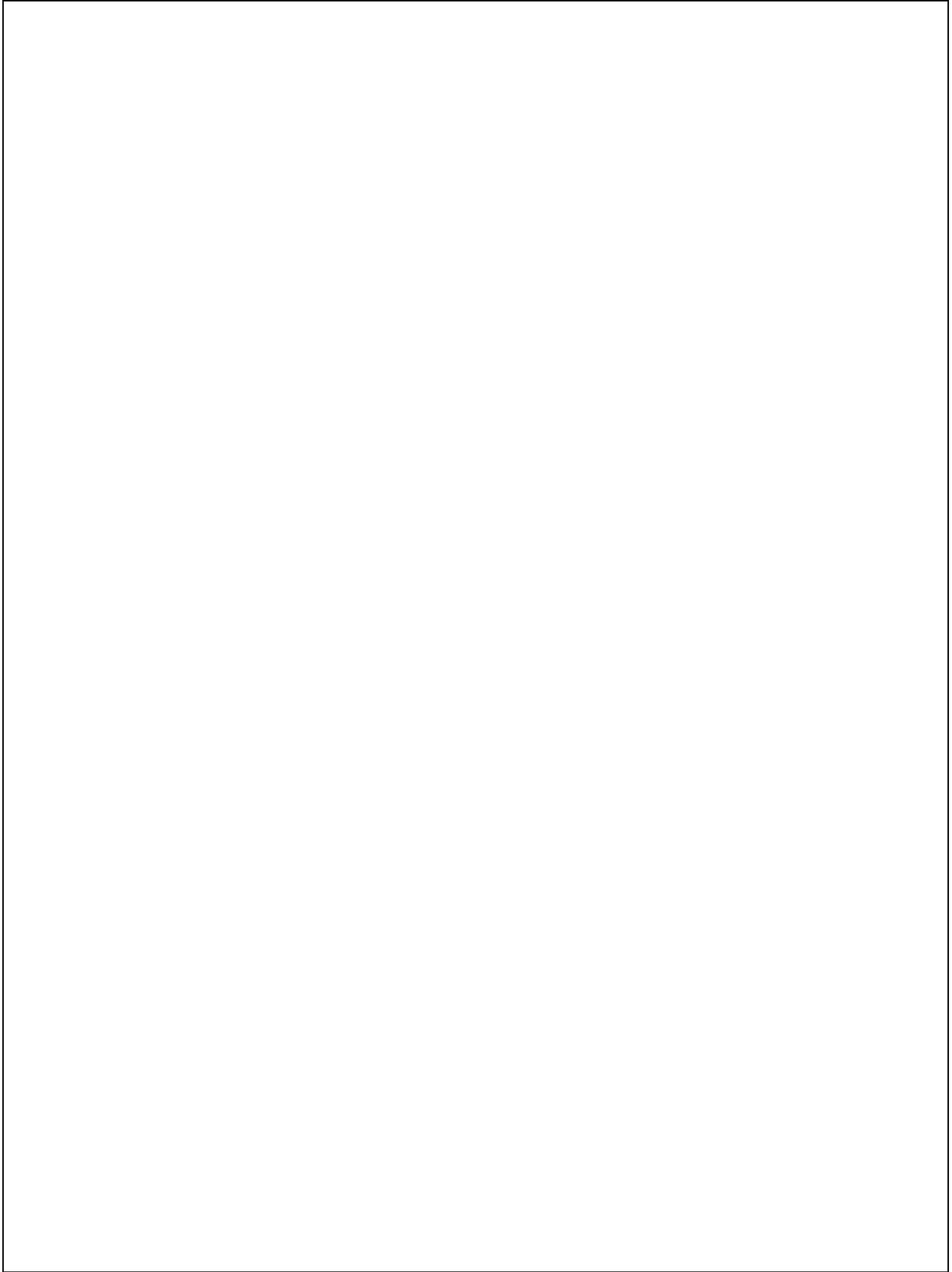
d) There are two main interfaces that extend the interface Collection, namely List and Set. Mention at least two differences between a List and a Set.

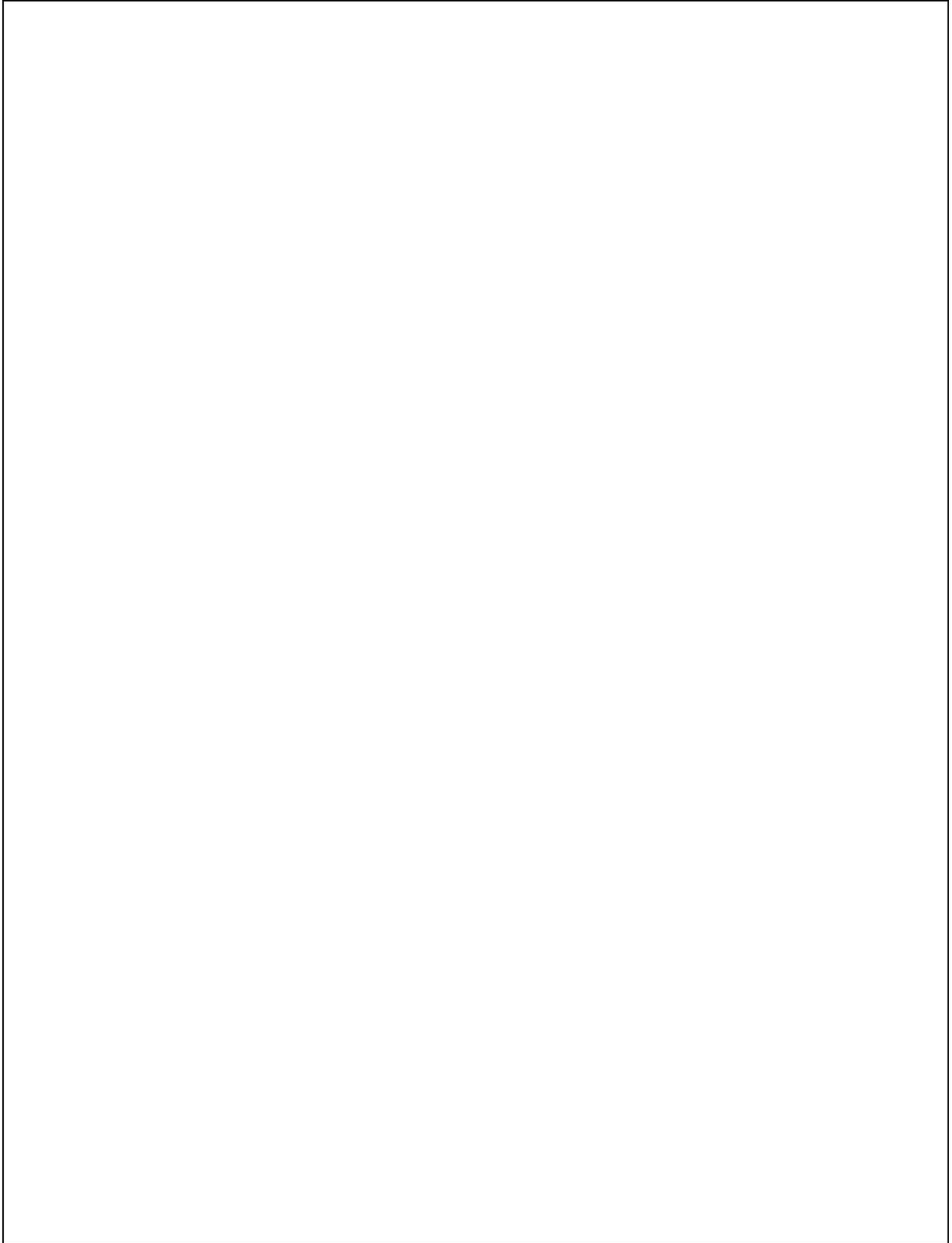
e) Explain why the List interface includes a method add with two parameters (element and index) while the Set interface does not.

Question 4 (Object-Oriented Concepts) [25 points]

Write a *FurniturePart* class and a *FurnitureKit* class. The *FurniturePart* class has three instance variables: part code, part name, and list of sub-parts if any. If a *FurniturePart* has no sub-parts, then part price should be an instance variable. The *FurnitureKit* class has code, name, color, and list of parts to make up a furniture kit and their count.

- a) Design the data part of both classes
- b) Write a method in the *FurnitureKit* class to calculate total price of the kit
- c) Write a method in the *FurnitureKit* class to display an itemized invoice that shows a list of all parts (with no sub-parts) and their quantity and cost per part





Question 5 (Searching and Sorting) [25 points]

- a) Class *Student* has instance variables “*ID: int*” and “*name: String*”. *Student* class implements *Comparable* interface. Provide the method definition for *compareTo* method of the *Student* class. A student can be compared based on his *ID*. Assume *equals* method is already implement for the student class and you can use it for your implementation of *compareTo* method. Using your implementation of *compareTo* provide a recursive *binarySearch* method that takes an array of students, a key (i.e. the student to search) and the left and right boundaries of the array. **[15 points]**

```
public class Student implements Comparable {  
  
    int ID;  
    String name;  
  
    public Student() {  
        ID = 0;  
        name = null;  
    }  
  
    public Student (int id, String str) {  
        ID = id;  
        name = str;  
    }  
  
}
```

```
public int compareTo(Object anObj) {  
//Provide Your Method Definition Here
```

```
}
```

```
public boolean equals(Object anObj) {
    if(anObj == null)
        return false;
    else if(this.getClass( ) != anObj.getClass( ))
        return false;
    else {
        Student otherStudent = (Student)anObj;
        return (otherStudent.ID == this.ID);
    }
}
```

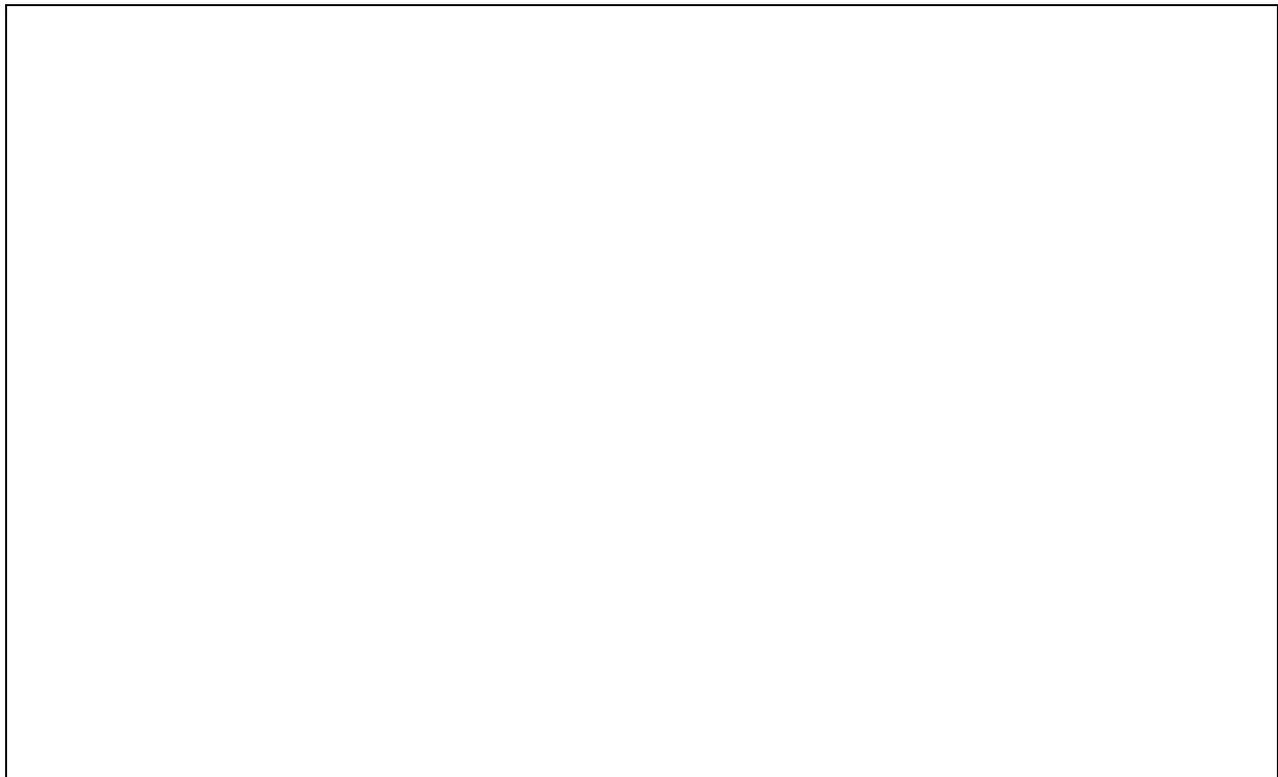
```
public static int binarySearch(Comparable target,Comparable[] a,
int left, int right) {
```

//Provide Your Method Definition Here

```
}
```

b) Display graphically the selection sort algorithm for the below array of integers. [5 points]

index	0	1	2	3	4	5	6
element	2	5	1	6	3	7	4



c) Display positioning (in steps) of the first pivot in quick sort algorithm. [5 points]

5	1	7	3	2	4	8	6
---	---	---	---	---	---	---	---

