Ex. No. 1.A **SEQUENTIAL SEARCH**

Date:

Problem :

Develop a Java application to Solve problems by using Sequential Search.

Aim:

To write a Java program to solve problems by using Sequential Search.

Algorithm:

Step1: start
Step 2: Get the array elements.
Step 3: Traverse the array.
Step 4: Match the key element with array element.
Step 5: If key element is found, return the index position of the array element.
Step 6: If key element is not found, print element not found message.
Step 7: stop

Program:

```
import java.util.Scanner;
class LinearSearch
{
  public static void main(String[] args)
  {
         int c,n,search,array[];

         Scanner in = new Scanner(System.in);
         System.out.println("Enter number of elements");
         n=in.nextInt();
         array=new int[n];

         System.out.println("Enter those “ + n + “ elements");

         for(c=0;c<n;c++)
           array[c]=in.nextInt();

         System.out.println("Enter value to find");
         search=in.nextInt();

         for(c=0;c<n;c++)
         {
              if(array[c]==search)
              {
                     System.out.println(search + " is present at location " + (c + 1) + ".");
                     break;
```

```
                }
            }
          if(c==n)
              System.out.println(search + " isn’t present in array.");
    }
}
```

Viva Questions:

1.How will you import the java packages?
2.What is the purpose of Linear Search?
3.How do you perform a Linear Search in array list?
4.How to implement Linear Search algorithm in java?
5.What is the complexity of Linear Search?

Result:

Thus the Java application to solve the problems by using Sequential Search was developed successfully.

Ex. No.1.B **BINARY SEARCH**

Date:

Problem :

Develop a Java application to Solve problems by using Binary Search.

Aim:

To write a Java program to solve problems by using Binary Search.

Algorithm:

```
Step1: Start
Step 2: Get the array elements.
Step 3: Calculate the mid element of the collection.
Step 4: Compare the key items with the mid element.
Step 5: If key item < mid element, then the key is in the upper half of the collection. Hence you
        need search in the upper half (mid +1).
Step 6: Else if key item = middle element, then we return the mid index position for the key
        found.
Step 7: Else key item > mid element, then the key lies in the lower half of the collection. Thus
        repeat binary search on the lower (right) half of the collection.
Step 8: Stop
```

Program:

```
import java.util.Scanner;
class BinarySearchExample
{
  public static void main(String args[])
  {
    int counter, num, item, array[], first, last, middle;
    Scanner input = new Scanner(System.in);
    System.out.println("Enter number of elements:");
    num = input.nextInt();
    array = new int[num];
    System.out.println("Enter " + num + " integers");
    for (counter = 0; counter < num; counter++)
        array[counter] = input.nextInt();

    System.out.println("Enter the search value:");
    item = input.nextInt();
    first = 0;
    last = num - 1;
    middle = (first + last)/2;

    while (first <= last)
    {
      if (array[middle] < item)
```

```
         first = middle + 1;
        else if(array[middle] == item)
         {
           System.out.println(item + " found at location " + (middle + 1) + ".");
           break;
         }
         else
         {
             last = middle - 1;
         }
         middle = (first + last)/2;
      }
      if ( first > last )
          System.out.println(item + " is not found.\n");
   }
}
```

Viva Questions:

1.How will you import the java packages?
2.What is array Binary Search?
3.How does arrays Binary Search work in java?
4.How to implement Binary Search algorithm in java?
5.What is the complexity of Binary Search?

Result:

Thus the Java application to solve the problems by using Binary Search was developed successfully.

Ex. No.1.C **QUADRATIC SORTING ALGORITHM**

Date: **SELECTION SORT**

Problem :

Develop a Java application to Solve problems by using Selection Sort.

Aim:

To write a Java program to solve problems by using Selection Sort.

Algorithm:

```
Step 1:  Start
Step 2:  Initialize minimum value (min_index) to location 0.
Step 3:  Traverse the array to find the minimum element in the array.
Step 4:  While traversing if any element smaller than min_index is found then swap both the
         values.
Step 5:  Then, increment min_index to point to the next element.
Step 6:  Repeat until the array is sorted.
Step 7:  Stop
```

Program:

```
import java.util.Scanner;
public class selectionsort
{
    public static void main(String args[])
    {
    int size, i, j, temp;
    int arr[] = new int[50];
    Scanner sc = new Scanner(System.in);
    System.out.print("Enter Array Size : ");
    size = sc.nextInt();
    System.out.print("Enter Array Elements : \n");
    for(i=0; i<size; i++)
    {
            arr[i] = sc.nextInt();
    }
    for(i=0; i<size; i++)
    {
            for(j=i+1; j<size; j++)
            {
                    if(arr[i] > arr[j])
                    {
                            temp = arr[i];
                            arr[i] = arr[j];
                            arr[j] = temp;
                    }
            }
    }
    System.out.print("sorted array >>\n");
    for(i=0; i<size; i++)
    {
            System.out.print(arr[i]+ " ");
    }
    }
}
```

Viva Questions:

1. What Is Selection Sort?
2. What is the worst case complexity of selection sort?
3. What is the advantage of selection sort over other sorting techniques?
4. What is the best case complexity of selection sort?
5. What are the disadvantages of Selection Sort?

Result:

Thus the java program for selection sort was developed and the output was verified successfully.

Ex. No.1.D **QUADRATIC SORTING ALGORITHM**

Date: **INSERTION SORT**

Problem:

Develop a java application to solve problems by using Insertion Sort.

Aim:

To write a Java program to solve problems by using Insertion Sort.

Algorithm:

```
Step1: Start
Step 2: Get the elements
Step 3: If the element is the first element, assume that it is already sorted.
Step 4: Pick the next element, and store it separately in a key.
Step 5: Now, compare the element with all elements in the sorted array.
Step 6: If the element in the sorted array is smaller than the current element, then move to the
        next element. Else, shift greater elements in the array towards the right.
Step 7: Insert the value.
Step 8: Repeat until the array is sorted.
Step 9: Stop
```

Program:

```java
import java.util.Scanner;
public class InsertionSort
{
  public static void main(String[] args)
  {
    int n, i, j, element;
    Scanner scan = new Scanner(System.in);
    System.out.print("Enter the Size of Array: ");
    n = scan.nextInt();
    int[] arr = new int[n];
    System.out.print("Enter " +n+ " Elements: ");
    for(i=0; i<n; i++)
      arr[i] = scan.nextInt();
    for(i=1; i<n; i++)
    {
      element = arr[i];
      for(j=(i-1); j>=0 && arr[j]>element; j--)
        arr[j+1] = arr[j];
        arr[j+1] = element;
    }
    System.out.println("\nThe new sorted array is: ");
    for(i=0; i<n; i++)
      System.out.print(arr[i]+ " ");
  }
}
```

Viva Questions:

1.How will import java package?
2.What is the purpose of Insertion Sort?
3.What is complexity of Insertion Sort?
4.Which sorting is best for large data?
5.How many comparisons does insertion sort have?

Result:

Thus the java program for insertion sort was developed and the output was verified successfully.

Ex. No.2.A **STACK**

Date:

Problem:

Develop stack data structure using classes and objects.

Aim:

To write a Java program to develop stack data structure using classes and objects.

Algorithm:

Step 1: start.

Step 2: Enter the size of stack.

Step 3: Enter the choice. Using switch case push and pop operations are performed.

Step 4: If the choice is one, then push operation is performed.

a) Enter the input of the element to be pushed.

b) Check whether the stack is full or not. If the stack is full, then print “stack overflow. Otherwise the value is pushed to the stack and the current position is increased.

Step 5: If the choice is two, then pop operation is performed.

a) Check whether the stack is empty or not. If the stack is empty, then print “stack underflow”.

b) Otherwise the value is popped from the stack.

Step 6: If the choice is three, display the stack elements.

Step 7: If the choice is four, exit the program.

Step 8: Stop.

Program:

```
import java.util.Scanner;
import java.util.*;
import java.io.*;
class Stack
{
        final int max=100;
        int s[]=new int[max];
        int top=-1;
        void push(int ele)
        {
                if(top>=max-1)
```

```
                    System.out.println("stack overflow");
                    else
                    s[++top]=ele;
}
int pop()
{
 int z=0;
if(top==-1)
          System.out.println("stack underflow");
          else
                    z=s[top--];
                    return z;
}
void display()
{
          if(top==-1)
                    System.out.println("Stack empty");
          else
          {
                    for(int i=top;i>-1;i--)
                    System.out.println(s[i]+"");
          }
}
public static void main(String args[])
{
int q=1;
Stack m = new Stack();
System.out.println("program to perform stack operations");
Scanner sc=new Scanner(System.in);
while(q!=0)
{
          System.out.println("enter 1.push 2.pop 3.display 4.exit");
          System.out.println("enter your choice");
          int ch=sc.nextInt();
          switch(ch)
          {
          case 1:
                    System.out.println("enter the element to be pushed");
                    int ele=sc.nextInt();
                    m.push(ele);
                    break;
          case 2:
                    int popele;
                    popele=m.pop();
                    System.out.println("the poped element is ");
                    System.out.println(popele+"");
                    break;
          case 3:
                    System.out.println("elements in the stack are");
                    m.display();
                    break;
            case 4:
                    q=0;
            }

        }

        }

}
```

Viva Questions:

1. Define stack?
2. Why Are Stacks Useful?
3. List out the basic operations that can be performed on a stack.
4. how to do push operation in a stack?
5. List the application of stacks.

Result:

Thus the Java program for stack data structure using classes and objects was developed and the output was verified successfully.

Ex. No.2.B **QUEUE**

Date:

Problem:

Develop Queue data structure using classes and objects.

Aim:

To implement queue data structure using classes and objects.

Algorithm:

```
1. Start
2. Define an array queue of size max = 5.
3. Initialize front = rear = –1.
4. Display a menu listing queue operations.
5. Accept choice
      If choice = 1 then enqueue operation is performed.
      If choice = 2 then dequeue operation is performed.
      If choice = 3 then display queue elements starting from front to rear.
      If choice = 4, exit the program.
7. Stop
```

Program:

```
import java.util.Scanner;
import java.util.*;
import java.io.*;
```

```
class Queue
{
        int SIZE = 5;
        int items[] = new int[SIZE];
        int front, rear;
        Queue()
        {
                front = -1;
                rear = -1;
        }
        boolean isFull()
        {
                if (front == 0 && rear == SIZE - 1)
                {
                        return true;
                }
                return false;
        }
        boolean isEmpty()
        {
                if (front == -1)
                return true;
                else
                return false;
        }
        void enQueue(int element)
        {
                if (isFull())
                {
                        System.out.println("Queue is full \n");
                }
                else
                {
                        if (front == -1)
                        front = 0;
                        rear++;
                        items[rear] = element;
                        System.out.println("\n Inserted " + element);
                }
        }
        int deQueue()
        {
                int element;
                if (isEmpty())
                {
                        System.out.println("\n Queue is empty \n");
                        return (-1);
                }
                else
```

```
		{
			element = items[front];
			if (front >= rear)
			{
				front = -1;
				rear = -1;
			} /* Q has only one element, so we reset the queue after deleting it. */
			else
			{
				front++;
			}
			System.out.println("\n Deleted -> " + element);
			return (element);
		}
	}
	void display()
	{
		/* Function to display elements of Queue */
		int i;
		if (isEmpty())
		{
			System.out.println("Empty Queue \n");
		}
		else
		{
			System.out.println("\nFront index-> " + front);
			System.out.println("Items -> ");
			for (i = front; i <= rear; i++)
			System.out.print(items[i] + " ");
			System.out.println("\nRear index-> " + rear);
		}
	}
	public static void main(String[] args)
	{
		int e=1;
		Queue q = new Queue();
		System.out.println("program to perform stack operations \n");
		Scanner sc=new Scanner(System.in);
		while(e!=0)
		{
			System.out.println("1.enqueue 2.dequeue 3.display 4.exit");
			System.out.println("\n enter your choice :");
			int ch=sc.nextInt();
			switch(ch)
			{
				case 1:
					System.out.println("\n enter the element to be pushed");
					int ele=sc.nextInt();
					q.enQueue(ele);
```

```
                                        break;
                                case 2:
                                        int dequeue;
                                        dequeue=q.deQueue();
                                        System.out.println("\n the deleted element is");
                                        System.out.println(dequeue+"");
                                        break;
                                case 3:
                                        System.out.println("\n elements in the queue are");
                                        q.display();
                                        break;
                                case 4:
                                        e=0;
                        }
                }
        }
}
```

Viva Questions:

1. Define a Queue?
2. List the application of queue.
3. What are the key public interfaces provided by queues?
4. What are the types of queue?
5. Define Dequeue?

Result:

Thus the Java program for Queue data structure using classes and objects was developed and the output was verified successfully.