DSA Lab Exam:-

Prn :-30

Name :- Rajesh Uttamrao Giri

Write a Java program to

a. Implement circular queue using arrays

```
package Circularque;



 class CircularQueue {
private int[] arr;
private int rear, front;
public CircularQueue(int size) {
arr = new int[size];
rear = -1;
front = -1;
}
public boolean isFull() {
return (front == -1 && rear == arr.length - 1) ||
(front == rear && front != -1);
}
public boolean isEmpty() {
return (front == rear && front == -1);
}
public void push(int val) {
if(isFull())
throw new RuntimeException("Queue is Full.");
rear = (rear + 1) % arr.length;
arr[rear] = val;
}
public void pop() {
if(isEmpty())
throw new RuntimeException("Queue is Empty.");
front = (front + 1) % arr.length;
if(front == rear) {
rear = -1;
front = -1;
}
}
public int peek() {
if(isEmpty())
throw new RuntimeException("Queue is Empty.");
int index = (front + 1) % arr.length;
return arr[index];
}
}
```

```
package Circularque;

import java.util.Scanner;

public class CircularQueueMain {
public static void main(String[] args) {
```

```
Scanner sc = new Scanner(System.in);
CircularQueue q = new CircularQueue(6);
int choice, val;
do {
System.out.println("\n0. Exit\n1. Push\n2. Pop\n3. Peek\nEnter choice: ");
choice = sc.nextInt();
switch(choice) {
case 1: // push
try {
System.out.print("Enter value to push: ");
val = sc.nextInt();
q.push(val);
} catch (Exception e) {
System.out.println(e.getMessage());
}
break;
case 2: // pop
try {
val = q.peek();
q.pop();
System.out.println("Popped: " + val);
} catch (Exception e) {
System.out.println(e.getMessage());
}
break;
case 3: // peek
try {
val = q.peek();
System.out.println("Peek: " + val);
} catch (Exception e) {
System.out.println(e.getMessage());
}
break;
}
}while(choice != 0);
sc.close();
}
}
```

```
0. Exit
1. Push
2. Pop
3. Peek
Enter choice:
1
Enter value to push: 10

0. Exit
1. Push
2. Pop
3. Peek
Enter choice:
1
Enter value to push: 20

0. Exit
1. Push
2. Pop
3. Peek
Enter choice:
2
Popped: 10
```

b. Perform quick sort to arrange given set of elements

```
package Quicksort;



import java.util.Arrays;
public class QuickSortMain {
public static void swap(int[] arr, int x, int y) {
int temp = arr[x];
arr[x] = arr[y];
arr[y] = temp;
}
public static void quickSort(int[] arr, int left, int right) {
// 0. if partition has single element or invalid partition, return.
if(left >= right)
return;
// consider left element as pivot -- arr[left]
int i=left, j=right;
while(i < j) {
// 1. from left (i-index) find element greater than pivot.
while(i <= right && arr[i] <= arr[left])
i++;
// 2. from right (j-index) find element less than or equal to pivot.
while(arr[j] > arr[left])
j--;
// 3. if i less than j, swap ith element with jth element
if(i < j)
swap(arr, i, j);
} // 4. repeat steps 1-3, till i < j
// 5. swap jth element with pivot element
swap(arr, j, left);
// 6. apply quick sort to left partition - left to j-1
quickSort(arr, left, j-1);
// 7. apply quick sort to right partition - j+1 to right
quickSort(arr, j+1, right);
```

```
}
public static void main(String[] args) {
int [] arr = {5, 3, 9, 1, 8, 7, 2, 6, 4};
// int[] arr = {4, 3, 2, 1};
System.out.println(Arrays.toString(arr));
quickSort(arr, 0, arr.length-1);
System.out.println(Arrays.toString(arr));
}
}
```