1. Write a program to implement stack using array. Implement functions for below operations.
a. Push element
b. Pop element
c. Peep element
d. Check if stack is full
e. Check if stack is empty

```
package genericStack;

import exception.ExceptionHandling;

public class Stack<T> {
    private int top;
    private T[] arr;

    public Stack(int size) {
        this.top = -1;
        arr = (T[]) new Object[size];
    }

    public Stack() {
        this.top = -1;
        arr = (T[]) new Object[10];
    }

    public boolean isEmpty() {
        if (top == -1)
            return true;
        return false;
    }

    public boolean isFull() {
        if (top == (arr.length - 1))
            return true;
        return false;
    }

    public void push(T data) throws ExceptionHandling {
        if (isFull())
            throw new ExceptionHandling("Full already");
        arr[++top] = data;
    }
```

```
    public T pop() throws ExceptionHandling {
        if (isEmpty())
            throw new ExceptionHandling("Empty alredy");
        return arr[top--];
    }

    public T peek() throws ExceptionHandling {
        if (isEmpty())
            throw new ExceptionHandling("Empty alredy");
        return arr[top];
    }
}
```

```
package question;

import java.util.Scanner;

import genericStack.Stack;

public class Question1 {

    public static void main(String[] args) {
        try(Scanner sc=new Scanner(System.in)){
            String option;
            Stack<Integer> stack=new Stack<>();
        do {
            System.out.println("Enter option a. Push element\r\n"
                    + "b. Pop element\r\n"
                    + "c. Peep element\r\n"
                    + "d. Check if stack is full\r\n"
                    + "e. Check if stack is empty");
            option=sc.next();
            switch (option) {
            case "a":
                stack.push(sc.nextInt());
                break;
            case "b":
                System.out.println(stack.pop());
                break;
            case "c":
                System.out.println(stack.peek());
```

```
                    break;
                case "d":
                    System.out.println(stack.isFull());
                    break;
                case "e":
                    System.out.println(stack.isEmpty());
                    break;
                }
            }while(option!="f");
        }catch (Exception e) {
            e.printStackTrace();
        }
        }

}
```

2. Write a program to reverse a string using stack

```
package question;

import java.util.Scanner;

import genericStack.Stack;

public class Question2 {

    public static void main(String[] args) {
        try (Scanner sc = new Scanner(System.in)) {
            Stack<Character> stack = new Stack<>();
            System.out.println("Enter String");
            String string = sc.next();
            int i = 0;
            while (i < string.length()) {
                stack.push(string.charAt(i));
                i++;
            }
            while (i-- > 0) {
                System.out.println(stack.pop());
            }
        } catch (Exception e) {
```

```
                e.printStackTrace();
        }

    }

}
```

3. Write a program to convert a decimal number into its binary form using stack.

```
package question;

import java.util.Scanner;

import genericStack.Stack;

public class Question3 {

    public static void main(String[] args) {
        try (Scanner sc = new Scanner(System.in)) {
            Stack<Integer> stack = new Stack<>();
            System.out.println("Enter any decimal");
            int number = sc.nextInt();
            while (number > 0) {
                int x = number % 2;
                number = number / 2;
                stack.push(x);
            }
            while (!stack.isEmpty()) {
                System.out.print(stack.pop());
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

}
```

4. Write a program to implement double ended stack.

```
class Dstack
{
int *arr;
int top1, top2, size;
};
```

In double ended stack addition and deletion of elements take place from both ends. Maintain two top indicators to perform operations.

```
package genericQueue;

import exception.ExceptionHandling;

public class DoubleEndedStack<T> {
   private int rear;
   private int front;
   private T[] arr;

   public DoubleEndedStack(int size) {
         super();
         this.rear = -1;
         this.front = -1;
         arr = (T[]) new Object[size];
   }

   public DoubleEndedStack() {
         super();
         this.rear = -1;
         this.front = -1;
         arr = (T[]) new Object[5];
   }

   public boolean isEmpty() {
         if (front > rear && rear == -1)
               return true;
         return false;
   }

   public boolean isFull() {
         if (front == (arr.length - 1))
               return true;
         return false;
   }
```

```
    public void enqueue(T data) throws ExceptionHandling {
        if (isFull())
            throw new ExceptionHandling("Full already");
        if (front == -1)
            front = 0;
        arr[++rear] = data;
    }

    public T dequeue() throws ExceptionHandling {
        if (isEmpty())
            throw new ExceptionHandling("Empty already");
        return arr[front++];
    }

    public void display() throws ExceptionHandling {
        if (isEmpty())
            throw new ExceptionHandling("Empty already");
        for (int i = front; i <= rear; i++)
            System.out.println(arr[i]);
    }
}
```

5. Write a program to convert the infix expression into its postfix form using stack. Accept infix string from user.
6. Write a program to convert infix expression into its prefix form using stack. Accept infix string from user.
7. Write a program to evaluate a postfix expression.
8. Write a program to implement queue using array. Implement functions for below operations.
a. Insert element in queue
b. Remove element from queue.
c. Print elements of queue.
d. Check if queue is full
e. Check if queue is empty.

```
package question;

import java.util.Scanner;

import genericQueue.DoubleEndedStack;
```

```
public class Question8 {

   public static void main(String[] args) {
         try (Scanner sc = new Scanner(System.in)) {
               DoubleEndedStack<Integer> queue = new DoubleEndedStack<>();
               String option;
               do {
                     System.out.println("Enter the option a. Insert element in queue\r\n"
                                 + "b. Remove element from queue.\r\n" + "c. Print
elements of queue.\r\n"
                                 + "d. Check if queue is full\r\n" + "e. Check if queue is
empty" + "f. Exit");
                     option = sc.next();
                     switch (option) {
                     case "a":
                           queue.enqueue(sc.nextInt());
                           break;

                     case "b":
                           System.out.println(queue.dequeue());
                           break;
                     case "c":
                           queue.display();
                           break;
                     case "d":
                           System.out.println(queue.isFull());
                           break;
                     case "e":
                           System.out.println(queue.isEmpty());
                           break;
                     }

               } while (option != "f");
         } catch (Exception e) {
               e.printStackTrace();
         }
   }

}
```

9. Reverse elements of stack using queue.

```
package question;

import java.util.Scanner;

import genericQueue.DoubleEndedStack;
import genericStack.Stack;

public class Question9 {

   public static void main(String[] args) {
         DoubleEndedStack<Character> queue =new DoubleEndedStack<>();
         Stack<Character> stack=new Stack<>();
         try(Scanner sc=new Scanner(System.in)){
               System.out.println("Enter string");
               String str=sc.next();
               int i=0;
               while(i<str.length()) {
                     stack.push(str.charAt(i));
                     i++;
               }
               while(i>0) {
                     queue.enqueue(stack.pop());
                     i--;
               }
               while(i<str.length()) {
                     System.out.println(queue.dequeue());
                     i++;
               }
         }catch (Exception e) {
               e.printStackTrace();
         }

   }

}
```

10.Implement circular queue using array with all operations mentioned in question 8.

11.Implement priority queue using array with all operations mentioned in question 8.
12.Write a program to implement double ended queue with array i.e. de-queue. Implement operations
a. Insert from front end
b. Insert from rear end
c. Remove from front end
d. Remove from rear end
e. Check if queue is full
f. Check if queue is empty
13.Write a menu driven program to create a singly linked list to perform following operations
a. Insert node at end, begin, middle of list
b. Remove node from end, begin, middle of list
c. Display list elements
14.Modify assignment 13 to print singly linked list in reverse way using recursion.
15.Implement doubly linked list with all operations mentioned in assignment 13
16.Implement singly circular linked list with all operations mentioned in assignment 13
17.Write a program to implement stack using linked list. Implement functions for below operations.
a. Push element
b. Pop element
c. Peep element
d. Check if stack is empty
18.Write a program to implement Binary Search Tree dynamically. Implement below operations
a. Insert a node with user entered value
b. Print inorder, preorder and postorder traversal of a tree
c. Delete a node with user entered value
d. Print height of a tree
19.Write a program to implement threaded binary tree with all operations mentioned in question 18
20.Write a program to implement AVL tree with all operations mentioned in question 18
21.Write a program to sort the elements of array using bubble sort
22.Write a program to sort the elements of array using selection sort
23.Write a program to sort the elements of array using insertion sort
24.Write a program to sort the elements of array using quick sort
25.Write a program to sort the elements of array using merge sort

26.Write a program to search an element inside array using linear search
27.Write a program to search an element inside array using binary search