Name: K.V.R saicharan
Roll no: 320022720012
1st semester
Sub: DS and Algorithms lab

12)a) Write a program for converting a given infix expression to postfix from using stack?
Aim:
Algorithm: •
Step 1 Scan the Infix Expression from left to right. •
Step 2 If the scanned character is an operand, append it with final Infix to Postfix string. •Step 3 : Else, •
Step 3.1 If the precedence order of the scanned(incoming) operator is greater than the precedence order of the operator in the stack (or the stack is empty or the stack contains a '(' or '[' or '{'), push it on stack. •
Step 3.2 Else, Pop all the operators from the stack which are greater than or equal to in precedence than that of the scanned operator. After doing that Push the scanned operator to the stack. (If you encounter parenthesis while popping then

stop there and push the scanned operator in the stack).

Step 4 If the scanned character is an '(' or '[' or '{', push it to the stack.

Step 5 If the scanned character is an ')' or ']' or '}', pop the stack and and output it until a '(' or '[' or '{' respectively is encountered, and discard both the parenthesis.

Step 6 Repeat steps 2-6 until infix expression is scanned.

Step 7 Print the output.

Step 8 Pop and output from the stack until it is not empty.

Program:

```
import java.util.Stack;
class Test{
static int Prec(char ch){
switch (ch){
case '+':
 case '-':
 return 1;
case '*':
case '/':
 return 2;
case
```

```
'^':
 return 3;
} return -1;  }
static String infixToPostfix(String exp) {
String result = new String("")
Stack stack = new Stack<>();
 for (int i = 0; i<exp.lenght ++i)
{  kar c = exp.charAt(i);
 if (Character.isLetterOrDigit(c))
 result+=c; else if (c == '('
stack.push(c);
 else if (c == ')') {
 while (!stack.isEmpty()
stack.peek()  =!'('
result += stack.pop();
 stack.pop();  }
else {
while (!stack.isEmpty() && Prec(c) <=
Prec(stack.peek())){
 result += stack.pop();
}stack.push(c);  }  }
while (!stack.isEmpty()){
 if(stack.peek() == '('
return "Invalid Expression"; result += stack.pop();  }
return result;
```

```
}
 public static void main(String[] args) { String exp =
"a+b*(c^d-e)^(f+g*h)-i";
System.out.println(infixToPostfix(exp)); } }
```

Output: abcd^e-fgh*+^*+i-

12)B.Write a program for sorting a list using Bubble sort and then apply binary search? Aim: Algorithm
:Step 1 Compare the element to check which one is greater
Step 2:Compare the second and third element to check which one is greater, and sort them in ascending order.
Step 3:Compare the third and fourth element to check which one is greater, and sort them in ascending order.
Step 4:Compare the fourth and fifth element to check which one is greater, and sort them in ascending order.
Step 5:Repeat steps 1-5 until no more swaps are required.
Program:
import

```
java.
import java.io.*;
class GFG { static void bubbleSort(int arr[], int n)
{ nt i, j,
temp; boolean swapped;
for (i = 0; i < n - 1; i++)

{ swapped = false;
for (j = 0; j < n - i - 1; j++) {
if (arr[j] > arr[j + 1]) {
temp = arr[j]; arr[j] = arr[j + 1]; arr[j + 1] = temp;
swapped = true; } }
if (swapped == false) break; } }
static void printArray(int arr[], int size) {
int i;
for (i = 0; i < size; i++)
System.out.print(arr[i] + " ");System.out.println
();  }
public static void main(String args[]) {
int arr[] = { 64, 34, 25, 12, 22, 11, 90 };
int n = arr.length; bubbleSort(arr, n);
System.out.println("Sorted array: ");
printArray(arr, n); } }
Output: Sorted array: 11 12 22 25 34 64
```