Government College of Engineering, Karad Programming for Problem Solving Lab

Nanekar Saurabh Rajesh

20141212

I1

**Experiment No. 8**

**Title**: Implement stack as an ADT to perform expression conversion and evaluation for postfix to prefix.

**Outcome:** Students can perform expression conversion and evaluation for postfix to prefix and its related applications.

**Theory:**

**Convert Postfix to Prefix Expression:**

**Example**:

Input: Postfix expression:  A B +

Output: Prefix expression- + A B

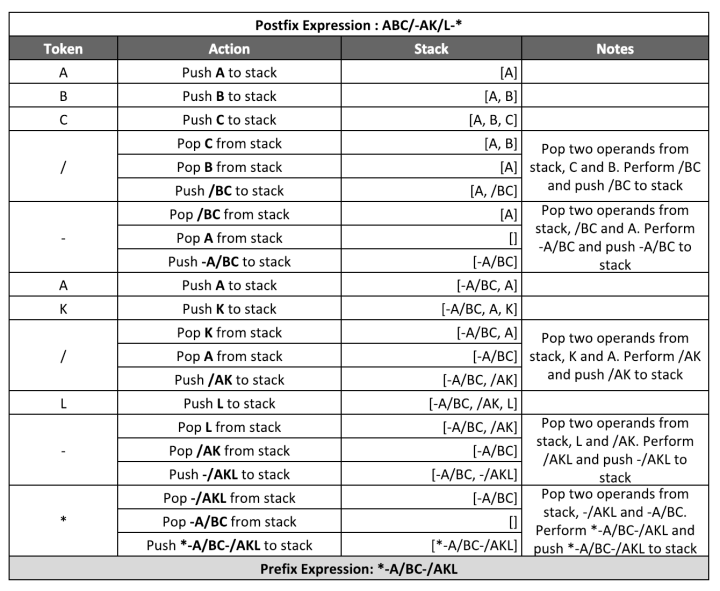
Input: Postfix expression:  ABC/-AK/L-\*

Output: Infix expression: \*-A/BC-/AKL

**Approach**: [Use Stack](https://algorithms.tutorialhorizon.com/stack-java-class-explained/" \t "_blank)

**Algorithm**:

1. Iterate the given expression from left to right, one character at a time
2. If the character is operand, push it to stack.
3. If the character is operator,
4. Pop operand from the stack, say it’s s1.
5. Pop operand from the stack, say it’s s2.
6. perform (operator s2 s1) and push it to stack.
7. Once the expression iteration is completed, initialize the result string and pop out from the stack and add it to the result.
8. Return the result.



**Analysis:**



**List of similar programs: Solve any one.**

1. Write a C program to solve Rat in a Maze problem using Backtracking technique in Stack data structure.

**Title Program:**Implement stack as an ADT to perform expression conversion and evaluation for postfix to prefix.

**Source code of Implemented Programs:**

//Nanekar Saurabh Rajesh\_20141212\_I1

#include<stdio.h>

#include<conio.h>

#include<string.h>

#include<stdlib.h>

#define MAX 20

char str[MAX], stack[MAX];

int top = -1;

void push(char c)

{

  stack[++top] = c;

}

char pop()

{

  return stack[top--];

}

int checkIfOperand(char ch)

{

  return (ch >= 'a' && ch <= 'z') || (ch >= 'A' && ch <= 'Z');

}

int isOperator(char x)

{

  switch (x) {

  case '+':

  case '-':

  case '/':

  case '\*':

    return 1;

  }

  return 0;

}

void postfixToprfix()

{

  int n, i, j = 0;

  char c[20];

  char a, b, op;

  printf("Enter the postfix expression\n");

  scanf("%s", str);

  n = strlen(str);

  for (i = 0; i < MAX; i++)

    stack[i] = '\0';

  printf("Prefix expression is:\t");

  for (i = n - 1; i >= 0; i--)

  {

    if (isOperator(str[i]))

    {

      push(str[i]);

    } else

    {

      c[j++] = str[i];

      while ((top != -1) && (stack[top] == '#'))

      {

        a = pop();

        c[j++] = pop();

      }

      push('#');

    }

  }

  c[j] = '\0';

  i = 0;

  j = strlen(c) - 1;

  char d[20];

  while (c[i] != '\0') {

    d[j--] = c[i++];

  }

  printf("%s\n", d);

}

int main()

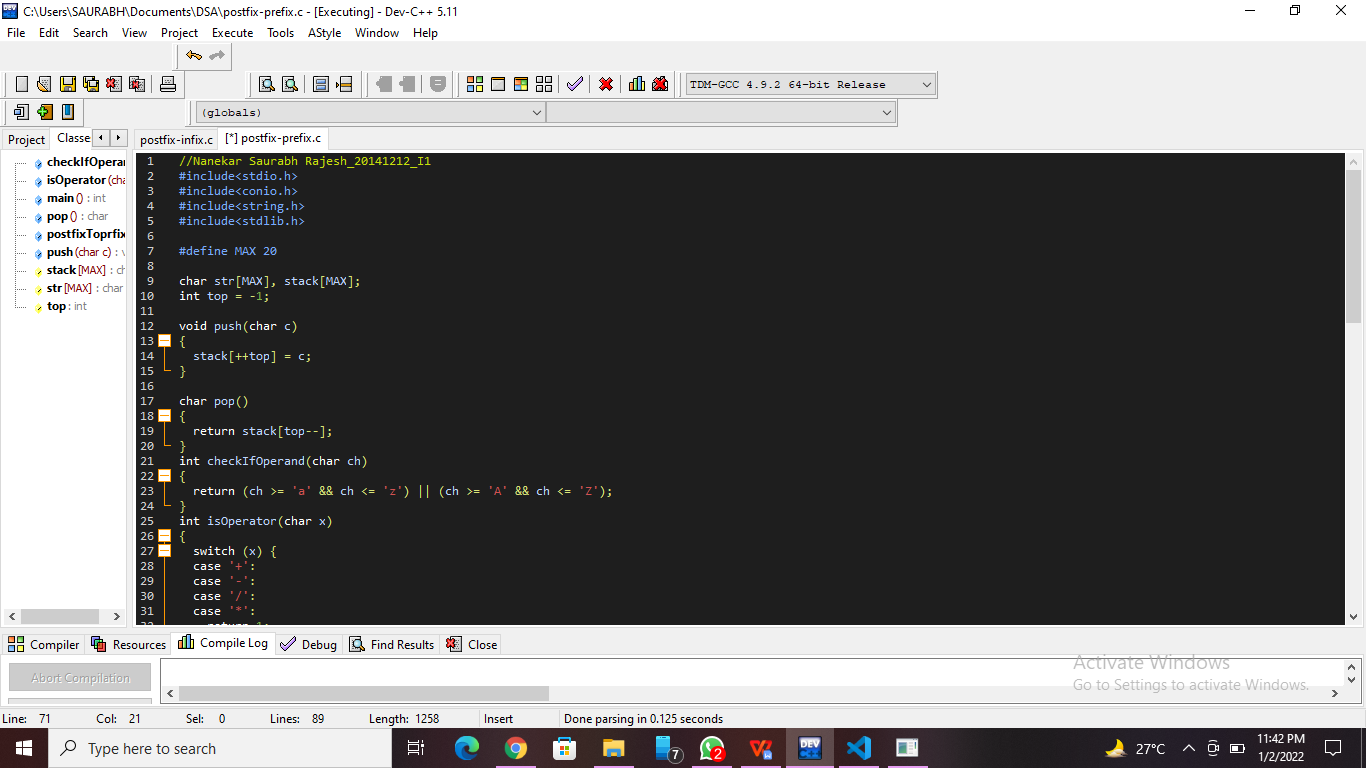
{

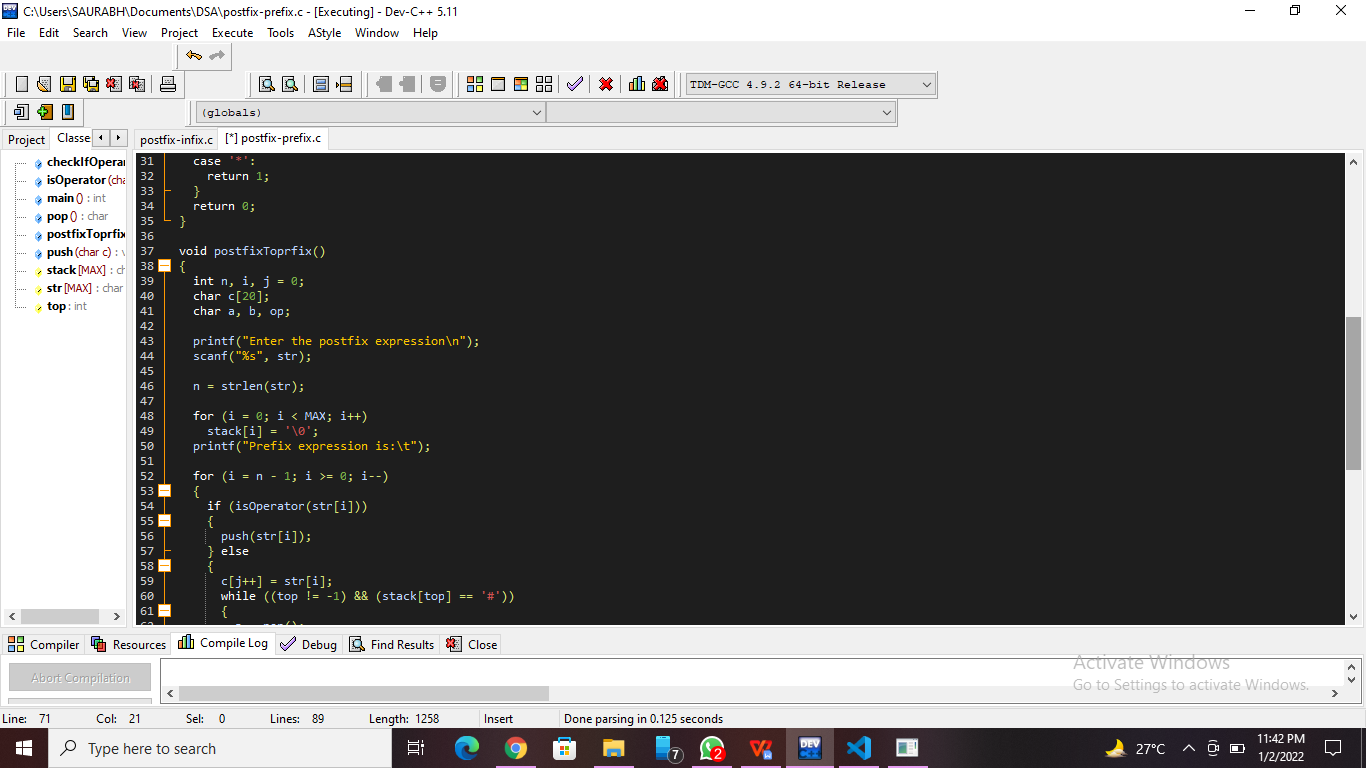
  postfixToprfix();

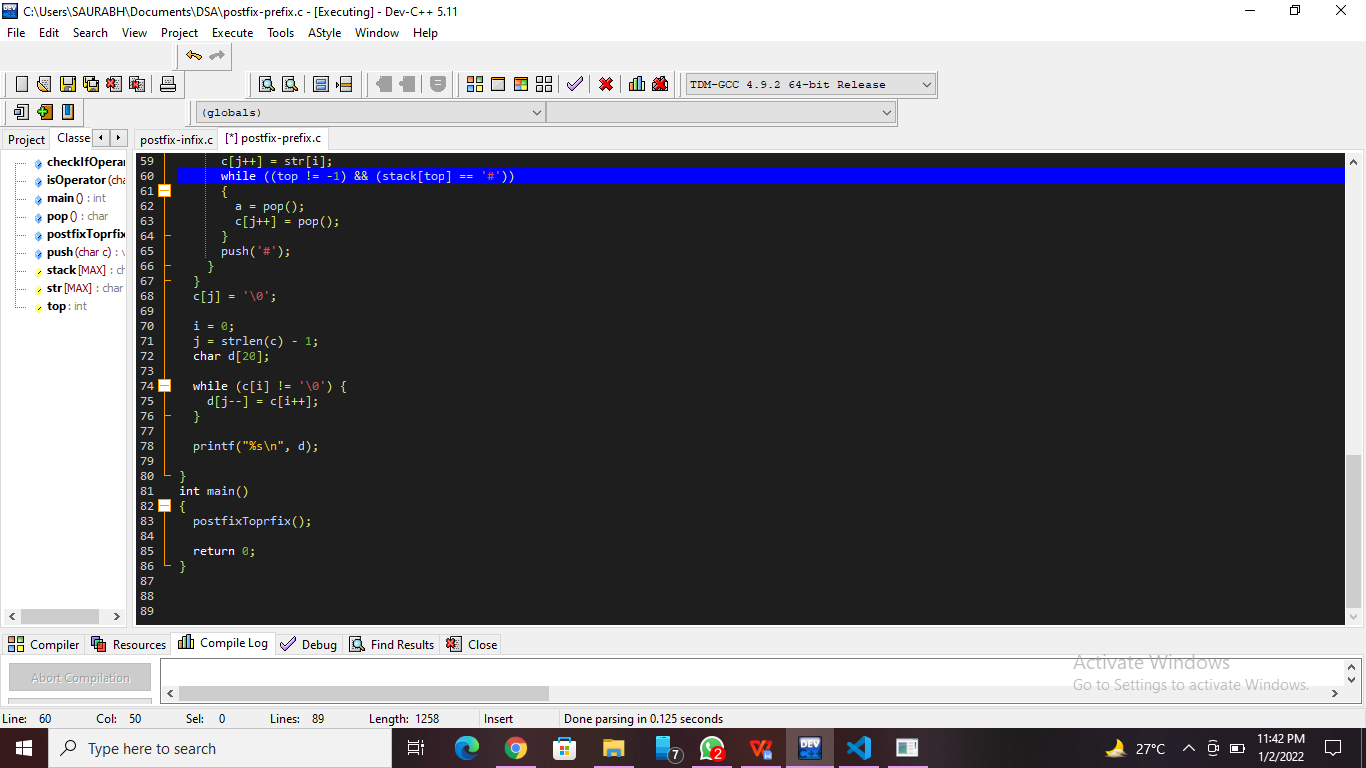
  return 0;

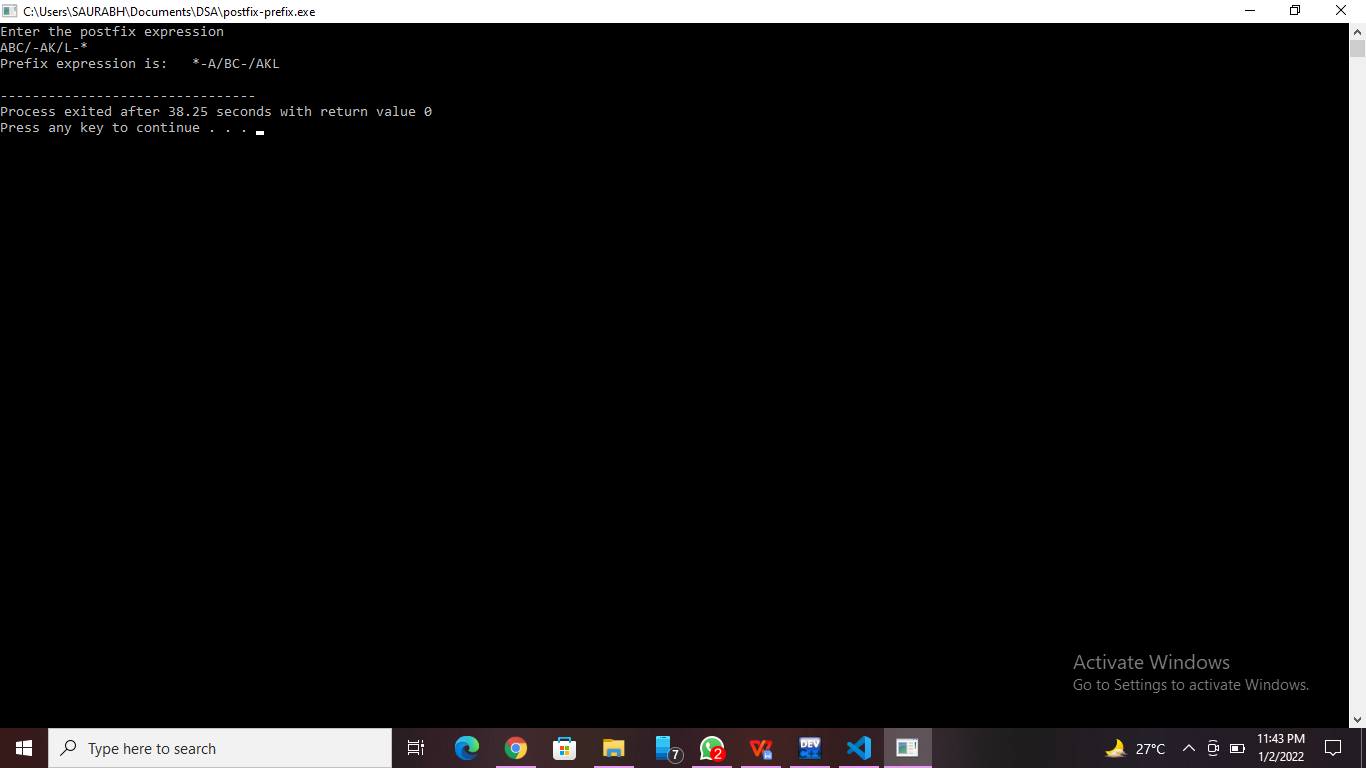
}

**Screenshots of Output:**

****

****

****

****

**Practice Program:**Write a C program to solve Rat in a Maze problem using Backtracking technique in Stack data structure.

//Nanekar Saurabh Rajesh\_20141212\_I1

#include <stdio.h>

#define SIZE 5

int maze[SIZE][SIZE] = {

    {0,1,0,1,1},

    {0,0,0,0,0},

    {1,0,1,0,1},

    {0,0,1,0,0},

    {1,0,0,1,0}

};

int solution[SIZE][SIZE];

void printsolution()

{

    int i,j;

    for(i=0;i<SIZE;i++)

    {

        for(j=0;j<SIZE;j++)

        {

            printf("%d\t",solution[i][j]);

        }

        printf("\n\n");

    }

}

int solvemaze(int r, int c)

{

    if((r==SIZE-1) && (c==SIZE-1))

    {

        solution[r][c] = 1;

        return 1;

    }

    if(r>=0 && c>=0 && r<SIZE && c<SIZE && solution[r][c] == 0 && maze[r][c] == 0)

    {

        solution[r][c] = 1;

        if(solvemaze(r+1, c))

            return 1;

        if(solvemaze(r, c+1))

            return 1;

        if(solvemaze(r-1, c))

            return 1;

        if(solvemaze(r, c-1))

            return 1;

        solution[r][c] = 0;

        return 0;

    }

    return 0;

}

int main()

{

    int i,j;

    for(i=0; i<SIZE; i++)

    {

        for(j=0; j<SIZE; j++)

        {

            solution[i][j] = 0;

        }

    }

    if (solvemaze(0,0))

        printsolution();

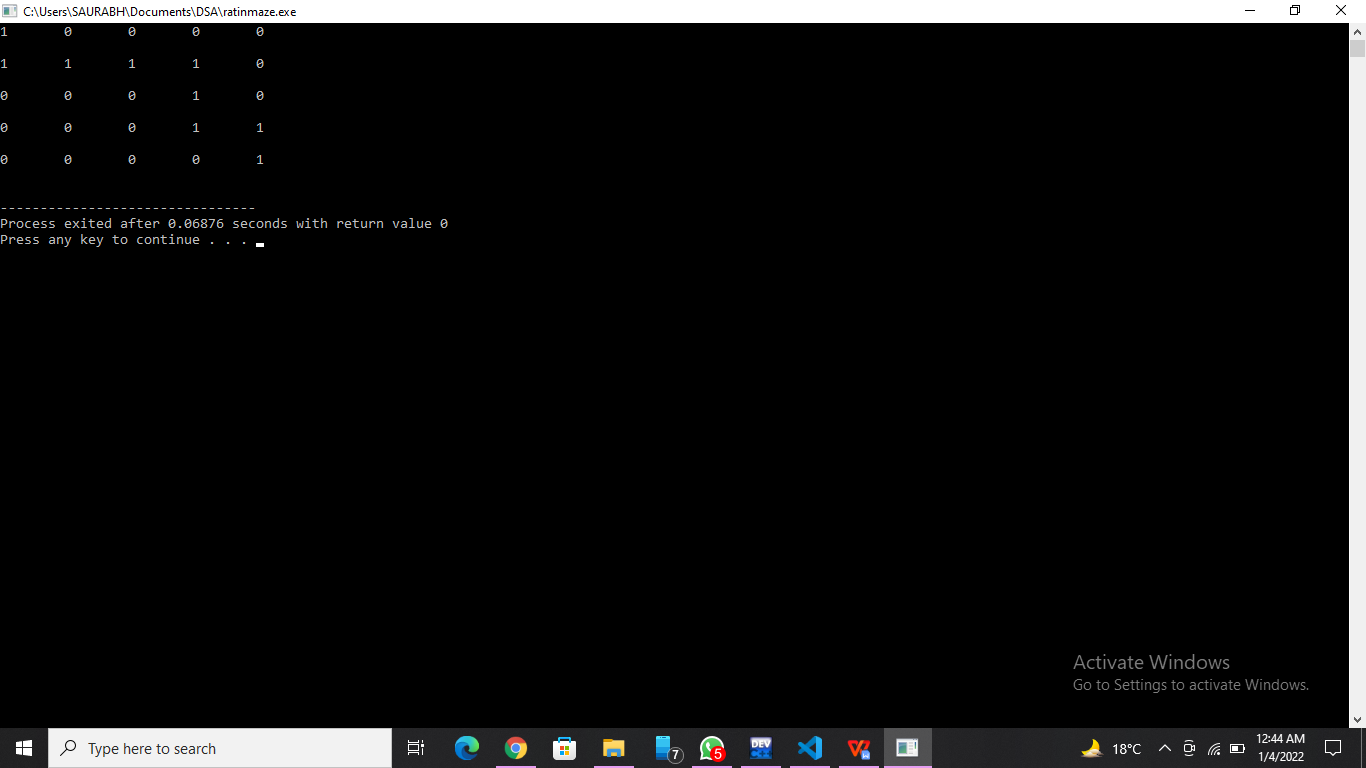
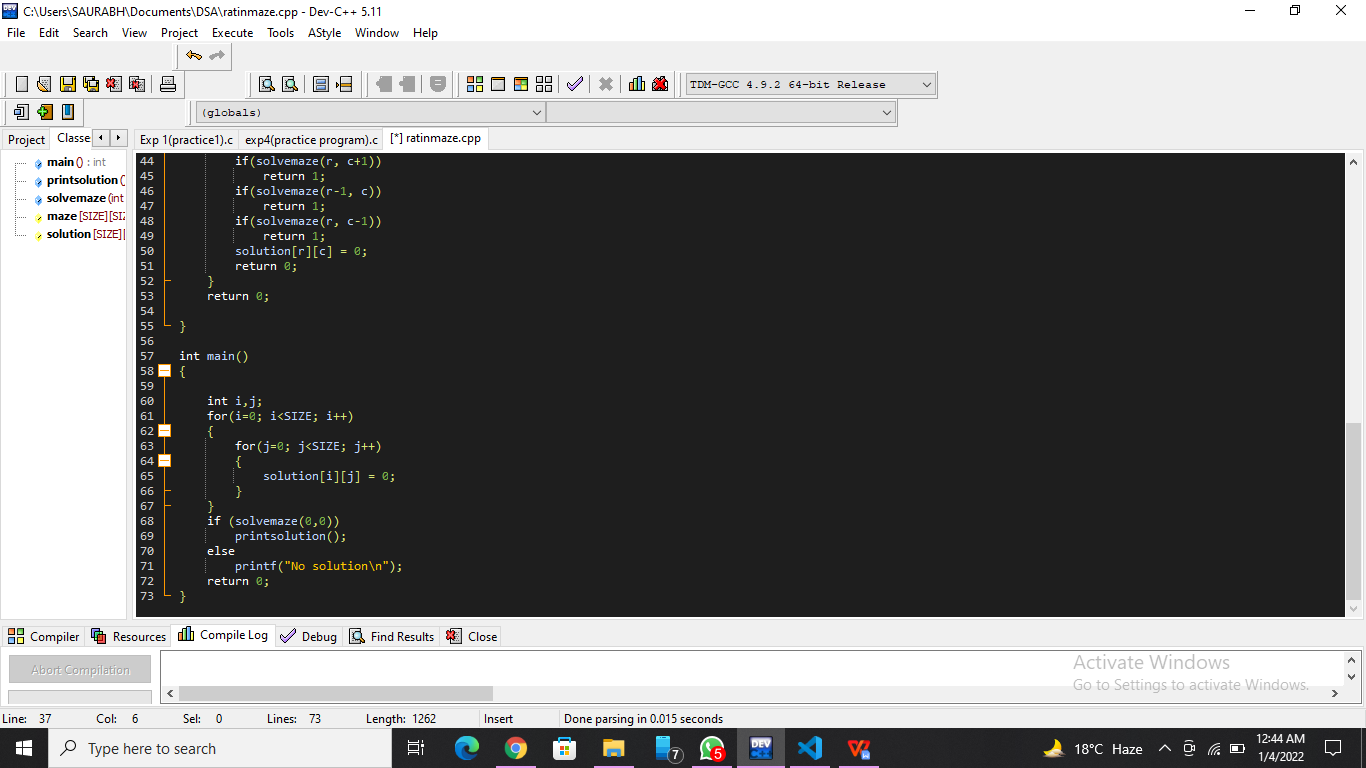
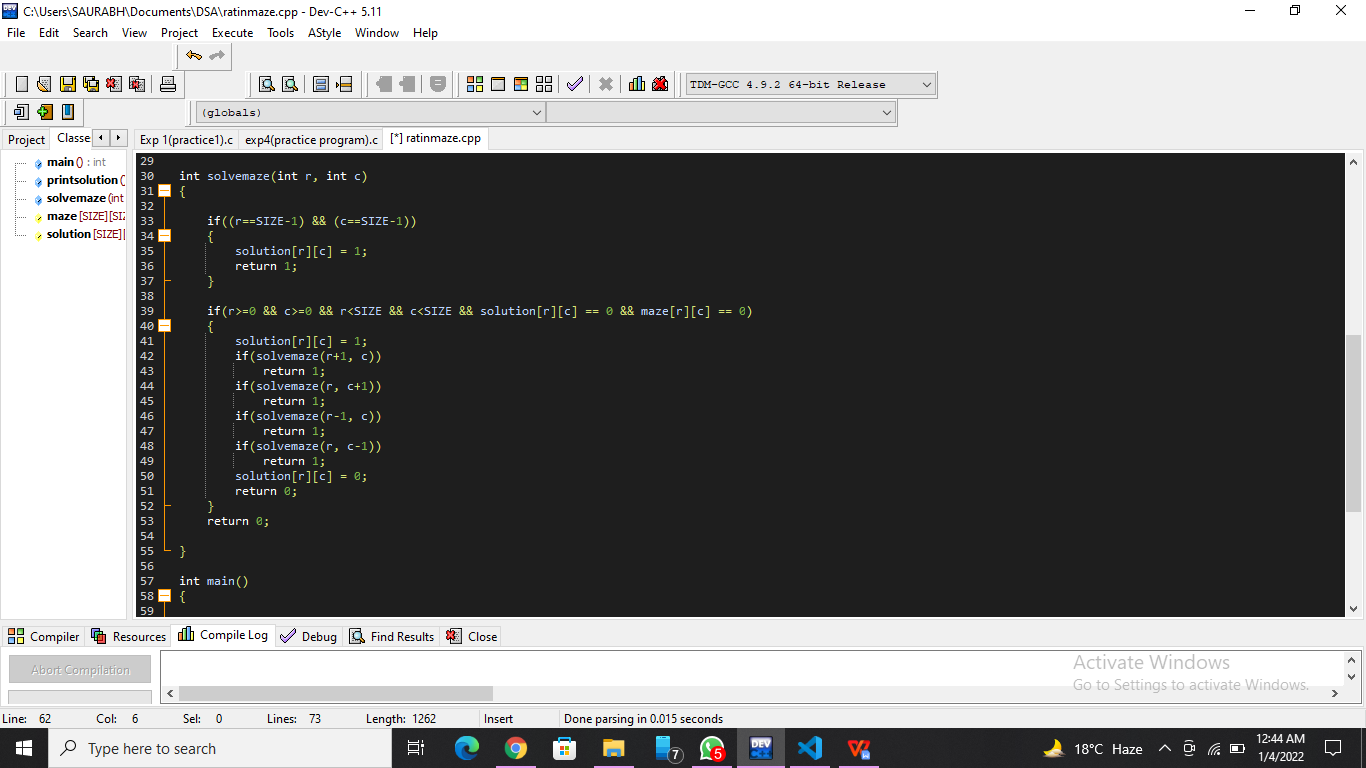
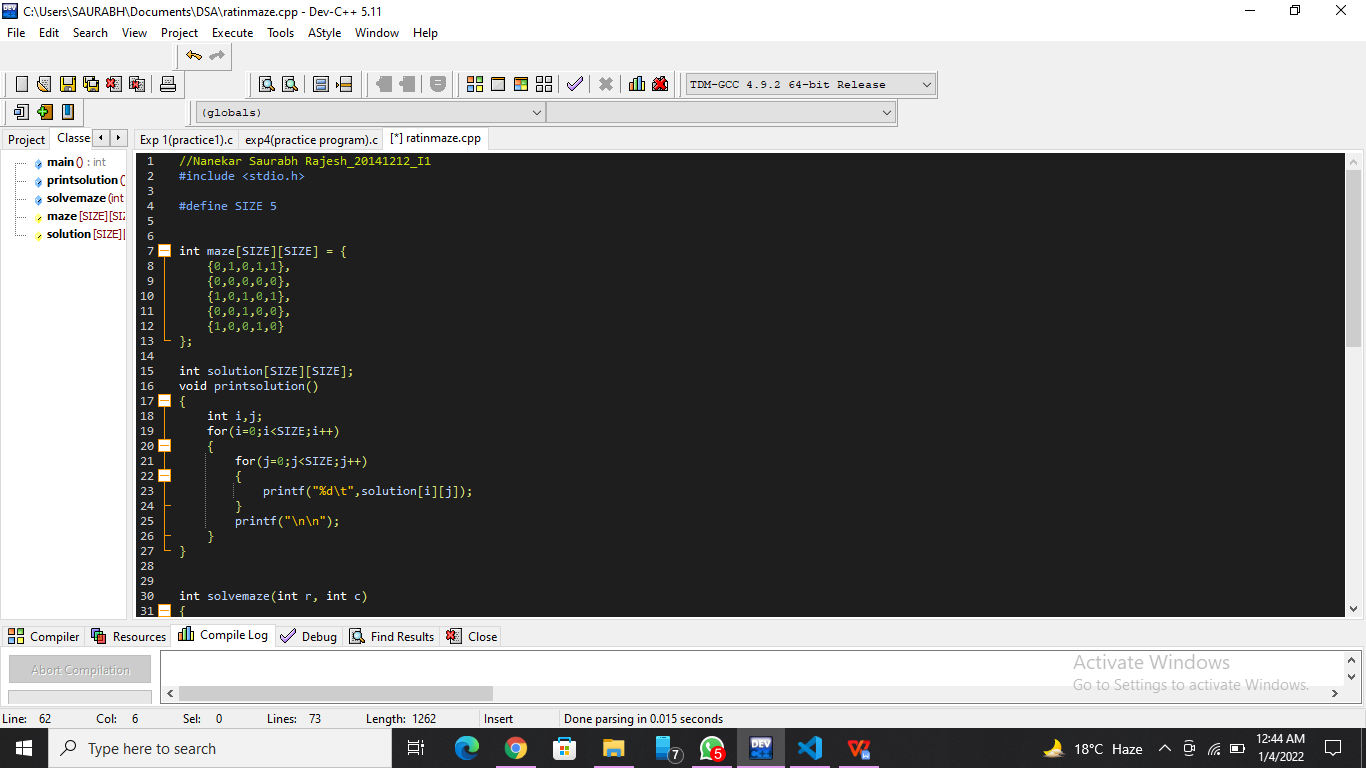
    else

        printf("No solution\n");

    return 0;

}

**Screenshots of Practice Program:-**

****

**List of sample questions for oral examination:**

1. What is backtracking in stack?
2. Is backtracking depth first search?
3. Why is postfix better than prefix?
4. What is prefix notation give an example?
5. What are advantages of prefixes?

**Conclusion:**

Learn to traverse a path.