**Experiment 2.3**

## Student Name: Vivek UID:

Branch: CSE Section/Group:

## Semester: 3rd Subject Name: Data Structures

Subject Code: 21CSH-211 Date of Performance: 29-09-22

## Aim/Overview of the practical:

Write a program to demonstrate the use of stack (implemented using linear array) in converting arithmetic expression from infix notation to postfix notation.

## 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,

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.

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.

1. Source Code: #include<iostream> using namespace std; char stack[50]; int top=-1; void push(char p)

{ stack[++top]=p;

} char pop()

{ if(top==-1) return - 1; else return stack[top--];

} int pr (char p)

{ if(p=='^')

return 3; if(p=='/' || p=='\*')

return 2; if(p=='+' || p=='-')

return 1;

return -1;

} int main()

{ char exp[100]; char \*a,p; cout<<"Enter the expression:"<<endl; cin>>exp; a=exp; while(\*a!='\0')

{ if(isalnum(\*a))

{ cout<<\*a; } else if(\*a=='(')

{ push(\*a); } else if(\*a==')')

{ while((p=pop()) != '(') cout<<p;

}

else

{ while(pr(stack[top])>=pr(\*a)) cout<<pop(); push(\*a); } a++;

} while(top!=-1)

{ cout<<pop();

} return 0;

}

1. Result/Output:



**Learning Outcomes:**

* Understood the concepts of stacks.
* Implemented push and pop commands.
* Learnt the concepts of infix and postfix.