**Exp. No. 8**

**Write a C program for stack implementation of Shift Reduce parser.**

**Program:**

#include <stdio.h>

#include <string.h>

#include <ctype.h>

#define MAX\_STACK\_SIZE 100

#define MAX\_INPUT\_LEN 100

char stack[MAX\_STACK\_SIZE];

int top = -1;

void push(char item)

{

if (top >= MAX\_STACK\_SIZE - 1)

{

printf("Stack overflow.\n");

return;

}

stack[++top] = item;

}

char pop()

{

if (top == -1)

{

printf("Stack underflow.\n");

return -1;

}

return stack[top--];

}

char peek()

{

if (top == -1)

return -1;

return stack[top];

}

int is\_operator(char symbol)

{

switch (symbol)

{

case '+':

case '-':

case '\*':

case '/':

return 1;

default:

return 0;

}

}

int precedence(char symbol)

{

switch (symbol)

{

case '+':

case '-':

return 1;

case '\*':

case '/':

return 2;

default:

return 0;

}

}

void shift\_reduce\_parser(char input[])

{

int i, j;

int len = strlen(input);

char symbol;

char temp[2];

for (i = 0; i < len; i++)

{

symbol = input[i];

if (isdigit(symbol))

{

printf("%c ", symbol);

}

else if (is\_operator(symbol) == 1)

{

while (precedence(stack[top]) >= precedence(symbol))

printf("%c ", pop());

push(symbol);

}

else if (symbol == '(')

{

push(symbol);

}

else if (symbol == ')')

{

while (stack[top] != '(')

printf("%c ", pop());

pop();

}

}

while (top > -1)

{

printf("%c ", pop());

}

}

int main()

{

char input[MAX\_INPUT\_LEN];

printf("Enter the expression: ");

scanf("%s", input);

shift\_reduce\_parser(input);

return 0;

}

**OUTPUT:**

