Practical 6

Implement a Stack and perform the stack operations: Infix to Postfix, Infix to Prefix, Evaluation of Postfix Expression, Print using Menu Driver Program such as 1.

Infix to Postfix , 2.

Infix to Prefix and 3.

Evaluation of Postfix Expression and 4. Exit.

```
#include <stdio.h>

#include <stdlib.h>

#include <ctype.h>

#include <string.h>

#define MAX 100

struct Stack

{

int top;

char items[MAX];

};

int isEmpty(struct Stack *stack)

{

return stack->top == -1;

}

char peek(struct Stack *stack)

{

return stack->items[stack->top];

}

char pop(struct Stack *stack)

{
```

```
if (!isEmpty(stack))
return stack->items[stack->top--];
return '\0';
}
void push(struct Stack *stack, char ch)
{
stack->items[++stack->top] = ch;
}
int precedence(char ch)
{
switch (ch)
{
case '+':
case '-':
return 1;
case '*':
case '/':
return 2;
case '^':
return 3;
}
return -1;
}
// Convert infix to postfix
void infixToPostfix(char *exp)
{
struct Stack stack;
stack.top = -1;
int i, k;
char output[MAX];
for (i = 0, k = 0; exp[i]; i++)
```

```
{
if (isalnum(exp[i]))
{
output[k++] = exp[i];
}
else if (exp[i] == '(')
{
push(&stack, exp[i]);
}
else if (exp[i] == ')')
{
while (!isEmpty(&stack) && peek(&stack) != '(')
output[k++] = pop(&stack);
pop(&stack); // Remove '('
}
else
{
while (!isEmpty(&stack) && precedence(exp[i]) <= precedence(peek(&stack)))
output[k++] = pop(&stack);
push(&stack, exp[i]);
}
}
while (!isEmpty(&stack))
output[k++] = pop(&stack);
output[k] = '\0';
printf("Postfix Expression: %s\n", output);
}
void reverse(char *exp)
{
int n = strlen(exp);
for (int i = 0; i < n / 2; i++)
```

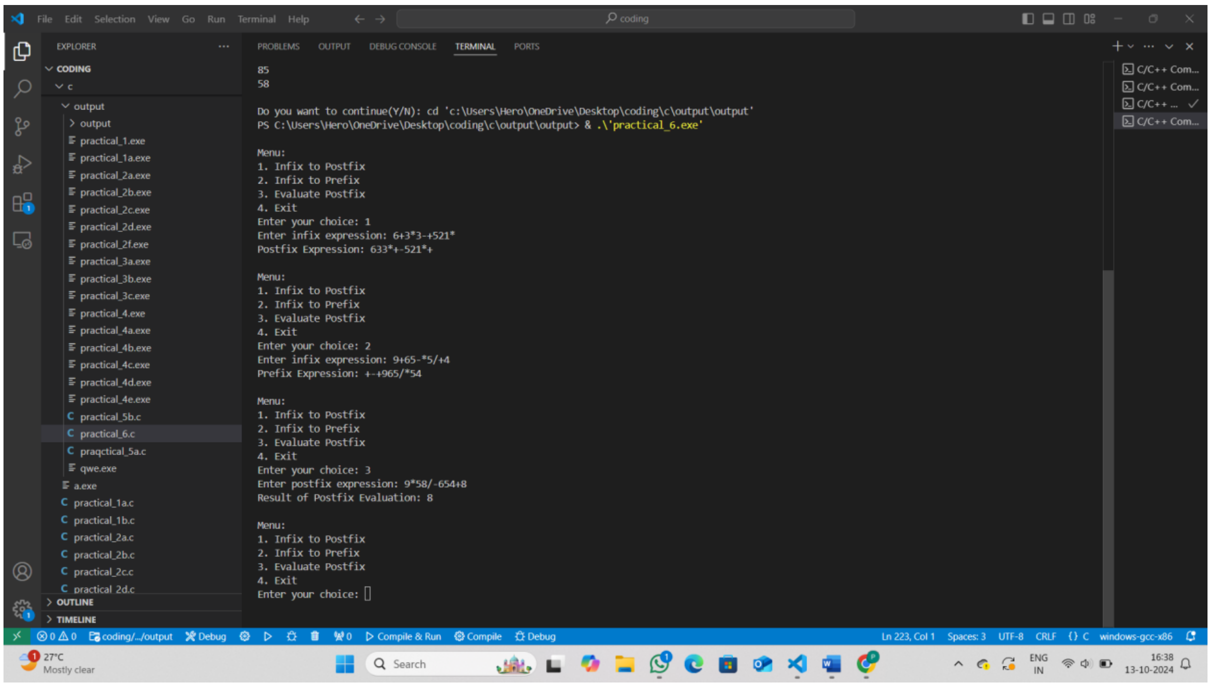
```
{
char temp = exp[i];
exp[i] = exp[n - i - 1];
exp[n - i - 1] = temp;
}
}
// Convert infix to prefix
void infixToPrefix(char *exp)
{
reverse(exp);
struct Stack stack;
stack.top = -1;
int i, k;
char output[MAX];
for (i = 0, k = 0; exp[i]; i++)
{
if (isalnum(exp[i]))
{
output[k++] = exp[i];
}
else if (exp[i] == ')')
{
push(&stack, exp[i]);
}
else if (exp[i] == '(')
{
while (!isEmpty(&stack) && peek(&stack) != ')')
output[k++] = pop(&stack);
pop(&stack);
}
else
```

```
{
while (!isEmpty(&stack) && precedence(exp[i]) < precedence(peek(&stack)))
output[k++] = pop(&stack);
push(&stack, exp[i]);
}
}
while (!isEmpty(&stack))
output[k++] = pop(&stack);
output[k] = '\0';
reverse(output);
printf("Prefix Expression: %s\n", output);
}
struct IntStack
{
int top;
int items[MAX];
};
int isIntStackEmpty(struct IntStack *stack)
{
return stack->top == -1;
}
int popInt(struct IntStack *stack)
{
return stack->items[stack->top--];
}
void pushInt(struct IntStack *stack, int value)
{
stack->items[++stack->top] = value;
}
// Evaluate postfix expression
int evaluatePostfix(char *exp)
```

```
{
struct IntStack stack;
stack.top = -1;
int i;
for (i = 0; exp[i]; i++)
{
if (isdigit(exp[i]))
{
pushInt(&stack, exp[i] - '0');
}
else
{
int val1 = popInt(&stack);
int val2 = popInt(&stack);
switch (exp[i])
{
case '+':
pushInt(&stack, val2 + val1);
break;
case '-':
pushInt(&stack, val2 - val1);
break;
case '*':
pushInt(&stack, val2 * val1);
break;
case '/':
pushInt(&stack, val2 / val1);
break;
}
}
}
```

```
return popInt(&stack);
}
int main()
{
int choice;
char exp[MAX];
do
{
printf("\nMenu:\n");
printf("1. Infix to Postfix\n");
printf("2. Infix to Prefix\n");
printf("3. Evaluate Postfix\n");
printf("4. Exit\n");
printf("Enter your choice: ");
scanf("%d", &choice);
getchar();
switch (choice)
{
case 1:
printf("Enter infix expression: ");
fgets(exp, MAX, stdin);
exp[strcspn(exp, "\n")] = 0;
infixToPostfix(exp);
break;
case 2:
printf("Enter infix expression: ");
fgets(exp, MAX, stdin);
exp[strcspn(exp, "\n")] = 0;
infixToPrefix(exp);
break;
case 3:
```

```c
printf("Enter postfix expression: ");

fgets(exp, MAX, stdin);

exp[strcspn(exp, "\n")] = 0;

printf("Result of Postfix Evaluation: %d\n", evaluatePostfix(exp));

break;

case 4:

printf("Exit the Program.\n");

exit(0);

break;

default:

printf("Invalid choice!\n");

}

} while (1);

return 0;

}
```



```
85
58

Do you want to continue(Y/N): cd 'c:\Users\Hero\OneDrive\Desktop\coding\c\output\output'
PS C:\Users\Hero\OneDrive\Desktop\coding\c\output\output> & .\'practical_6.exe'

Menu:
1. Infix to Postfix
2. Infix to Prefix
3. Evaluate Postfix
4. Exit
Enter your choice: 1
Enter infix expression: 6+3*3-+521*
Postfix Expression: 633*+-521*+

Menu:
1. Infix to Postfix
2. Infix to Prefix
3. Evaluate Postfix
4. Exit
Enter your choice: 2
Enter infix expression: 9+65-*5/+4
Prefix Expression: +-+965/*54

Menu:
1. Infix to Postfix
2. Infix to Prefix
3. Evaluate Postfix
4. Exit
Enter your choice: 3
Enter postfix expression: 9*58/-654+8
Result of Postfix Evaluation: 8

Menu:
1. Infix to Postfix
2. Infix to Prefix
3. Evaluate Postfix
4. Exit
Enter your choice: 
```