Batch: B1 **Experiment Number:3**

Roll Number: 16010420133 **Name:Soumen samanta**

---

a) Aim of the Experiment: WAP to create a stack using SLL by implementing following operations
1. Create stack, 2. Insert an element, 3. Delete an element, 4. Display top element.

b) WAP to convert a given infix expression into equivalent postfix form using stack implemented in part (a).

Program/ Steps:

```
#include
<stdio.h>
            #include <stdlib.h>
            #include <math.h>
            void partA();
            void partB();
            void createStack();
            void push();
            void pop();
            void display();
            int IsEmpty(int var);
            void infixToPostfix();

            int isOperand(char ch);
            int priority(char sym);
            char popChar();
            void calc(char ch);
            void pushNum();
            char peek();
            void pushChar(char val);
            void pushNum(int val);
            int popNum();
            int peekNum();

            struct stack
            {
                int val;
```

```
    struct stack *next;
} * top;
struct stack2
{
    char ch;
    struct stack *next;
} * characters;
int main()
{
    int ch = 0;
    printf("\n===== Stack operations using linked list =======\n");
    while (ch != 3)
    {
        printf("\n1. Part A");
        printf("\n2. Part B");
        printf("\n3. Exit");
        printf("\nEnter your choice from 1-3: ");
        scanf("%d", &ch);
        switch (ch)
        {
        case 1:
            partA();
            break;
        case 2:
            partB();
            break;
        case 3:
            break;
        default:
            printf("\nEnter your choice from 1-3 only");
            break;
        }
    }
    printf("\nThank you");
    return 0;
}
void partA()
{
    int choice = 0;
    while (choice != 6)
    {
        printf("\n\nChose one from the below options...\n");
        printf("\n1.Create Stack\n2.Push\n3.Pop\n4.Peek\n5.IsEmpty\n6.Back");
```

```
printf("\nEnter your choice from 1-6: ");
scanf("%d", &choice);
switch (choice)
{
case 1:
{
    createStack(1);
    printf("New stack created");
    break;
}
case 2:
{
    push();
    break;
}
case 3:
{
    pop();
    break;
}
case 4:
{
    display();
    break;
}
case 5:
{
    if (IsEmpty(1))
    {
        printf("\nTrue\n");
    }
    else
    {
        printf("\nFalse\n");
    }
    break;
}
case 6:
{
    break;
}
default:
{
```

```
            printf("Please Enter valid choice from 1-6 only");
        }
        };
    }
}
void createStack(int stackType)
{
    if (stackType == 1)
    {
        if (top != NULL)
            while (top != NULL)
                pop();
    }
    else
    {
        if (characters != NULL)
            while (characters != NULL)
                popChar();
    }
}
void push()
{
    int val;
    struct stack *ptr = (struct stack *)malloc(sizeof(struct stack));
    if (ptr == NULL)
    {
        printf("not able to push the element");
    }
    else
    {
        printf("Enter the value: ");
        scanf("%d", &val);
        if (top == NULL)
        {
            ptr->val = val;
            ptr->next = NULL;
            top = ptr;
        }
        else
        {
            ptr->val = val;
            ptr->next = top;
            top = ptr;
```

```
        }
        printf("Item pushed");
    }
}

void pop()
{
    int item;
    struct stack *ptr;
    if (top == NULL)
    {
        printf("Underflow");
    }
    else
    {
        item = top->val;
        ptr = top;
        top = top->next;
        free(ptr);
        printf("Item popped");
    }
}
int IsEmpty(int type)
{
    if (type == 1)
        return top == NULL;
    else
        return characters == NULL;
}
void display()
{
    int i;
    struct stack *ptr;
    ptr = top;
    if (ptr == NULL)
    {
        printf("Stack is empty\n");
    }
    else
    {
        printf("Printing Stack elements \n");
        printf("%d\n", ptr->val);
    }
```

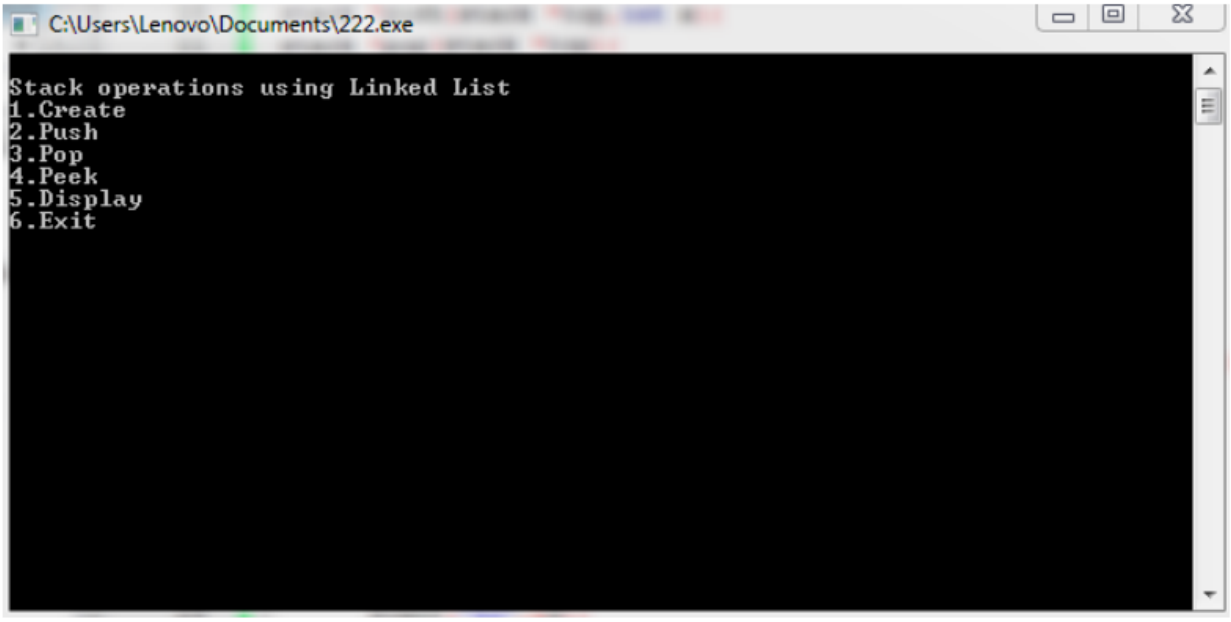
```
}

// --------------Part B--------------
void partB()
{

    createStack(2);
    infixToPostfix();
}
void infixToPostfix()
{
    char exp[50];
    printf("\nEnter the expression: ");
    scanf("%s", exp);
    printf("\nPostfix equation: ");
    char ch, stringNum[10] = {'\0'}, temp;
    int pos = 0;
    for (int i = 0; i < strlen(exp); i++)
    {
        ch = exp[i];
        if (isOperand(ch))
        {
            strncat(stringNum, &exp[i], 1);
        }
        else
        {
            if (stringNum[0] != '\0')
            {
                int num = atoi(stringNum);
                printf("%d ", num);
                pushNum(num);
                stringNum[0] = '\0';
            }
        check:
            if (!IsEmpty(2) && priority(ch) == 0)
            {
                pushChar(ch);
            }
            else if (ch == ')')
            {
                while (peek() != '(')
                {
                    temp = popChar();
```

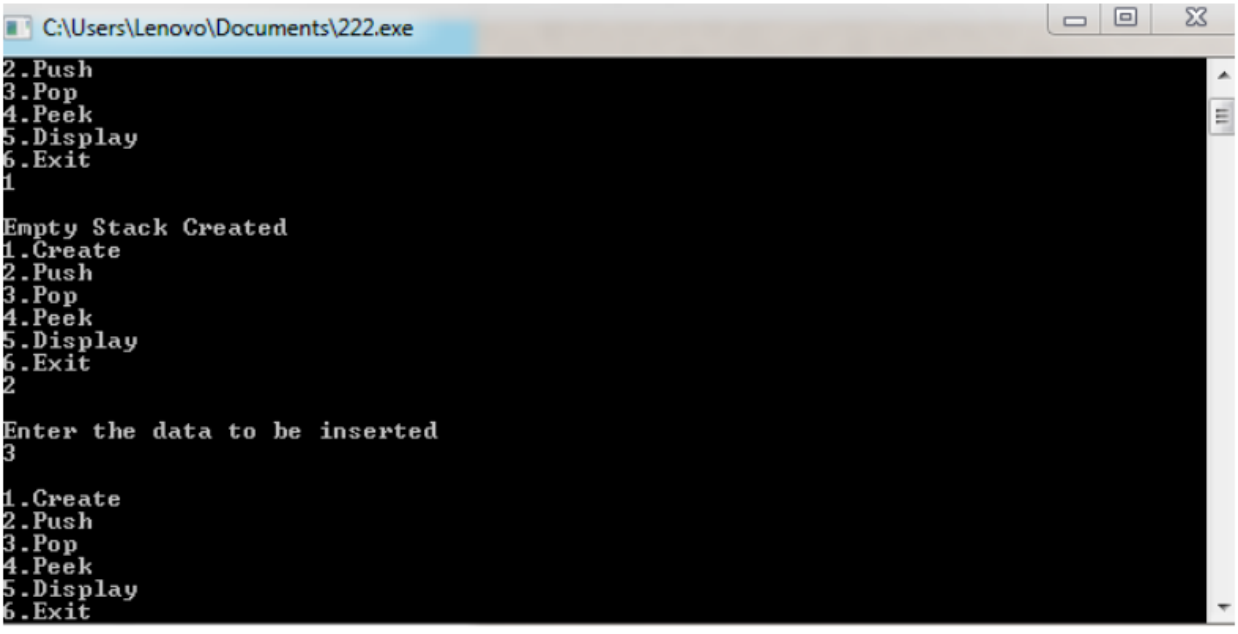
```
                    printf("%c ", temp);
                    calc(temp);
                }
                popChar();
            }
            else if (!IsEmpty(2) && priority(peek()) >= priority(ch))
            {
                temp = popChar();
                printf("%c ", temp);
                calc(temp);
                goto check;
            }
            else
            {
                pushChar(ch);
            }
        }
    }
    if (stringNum[0] != '\0')
    {
        int num = atoi(stringNum);
        printf("%d ", num);
        pushNum(num);
        stringNum[0] = '\0';
    }
    while (!IsEmpty(2))
    {
        temp = popChar();
        printf("%c ", temp);
        calc(temp);
    }
    printf("\nResult: %d\n", top->val);
}

void calc(char ch)
{
    int a, b;
    b = popNum();
    a = popNum();
    switch (ch)
    {
    case '+':
        pushNum(a + b);
```

```
        break;
    case '-':
        pushNum(a - b);
        break;
    case '*':
        pushNum(a * b);
        break;
    case '/':
        pushNum((int)(a / b));
        break;
    case '^':
        pushNum((int)pow((double)a, (double)b));
        break;
    }
}

int isOperand(char ch)
{
    return (ch >= '0' && ch <= '9');
}
int priority(char symbol)
{
    switch (symbol)
    {
    case '(':
        return 0;
    case '+':
    case '-':
        return 1;
    case '*':
    case '/':
        return 2;
    case '^':
        return 3;
    }
}
void pushChar(char val)
{
    struct stack2 *ptr = (struct stack2 *)malloc(sizeof(struct stack2));
    ptr->ch = val;
    ptr->next = characters;
    characters = ptr;
}
```

```
char popChar()
{
    char var;
    struct stack2 *ptr;
    var = characters->ch;
    ptr = characters;
    characters = characters->next;
    free(ptr);
    return var;
}

char peek()
{
    return characters->ch;
}

void pushNum(int val)
{
    struct stack *ptr = (struct stack *)malloc(sizeof(struct stack));
    ptr->val = val;
    ptr->next = top;
    top = ptr;
}
int popNum()
{
    int var;
    struct stack *ptr;
    var = top->val;
    ptr = top;
    top = top->next;
    free(ptr);
    return var;
}
int peekNum()
{
    return top->val;
}
```

Output/Result:





```
1. Part A
2. Part B
3. Exit
Enter your choice from 1-3: 2

Enter the expression: (2*5-1*2)/(11-9)

Postfix equation: 2 5 * 1 2 * - 11 9 - /
Result: 4

1. Part A
2. Part B
3. Exit
Enter your choice from 1-3: Killed
```

CONCLUSION

Stack is a linear data structure which follows a particular order in which the operations are performed. The order may be LIFO(Last In First Out) or FILO(First In Last Out).

One of the use of stack data structure, conversion of infix expression to postfix expression. By scanning the infix expression from left to right, when we will get any operand, simply add them to the postfix form, and for the operator and parenthesis, add them in the stack maintaining the precedence of them.

OUTCOME

Explain different data structures used in problem solving. Apply linear data structure.

References:

Books/ Journals/ Websites:

- Y. Langsam, M. Augenstin and A. Tannenbaum, “Data Structures using C”, Pearson Education Asia, 1st Edition, 2002