Shrihari K Mangle
CSB2-74 12010517

LAB-4

4. A. Implement stack using Linked Lists.

   B. Implement the Infix expression to Postfix and Prefix conversion using Stack.

A. STACK USING LL

CODE :

```c
#include <stdio.h>
#include <stdlib.h>

struct Node
{
    int data;
    struct Node *next;
};

struct Node *top = NULL;

int isEmpty(struct Node *top)
{
    if (top == NULL)
    {
        return 1;
    }

    return 0;
}

int isFull(struct Node *top)
{
    struct Node *p = (struct Node *)malloc(sizeof(struct Node));

    if (p == NULL)
    {
        return 1;
    }

    return 0;
}

struct Node *push(struct Node *top, int value)
{
    if (isFull(top))
    {
        printf("Stack Overflow\n");
    }
    else
    {
        struct Node *n = (struct Node *)malloc(sizeof(struct Node));
```

```
        n->data = value;
        n->next = top;
        top = n;

        return top;
    }
}

int pop()
{
    if (isEmpty(top))
    {
        printf("Stack Underflow\n");
    }
    else
    {
        struct Node *n = top;
        int x = top->data;
        top = top->next;
        free(n);

        return x;
    }
}

int stackTop()
{
    return top->data;
}

int stackBottom(struct Node *top)
{
    struct Node *ptr = top;
    while (ptr->next != NULL)
    {
        ptr = ptr->next;
    }
    return ptr->data;
}

void StackDisplay(struct Node *ptr)
{
    struct Node *temp = ptr;
    while (temp != NULL)
    {
        printf("%d -> ", temp->data);
        temp = temp->next;
    }
    printf("NULL\n\n");
}

int main()
{
    int choice, no;

    while (1)
    {
```

```
        printf("ENTER THE CHOICE : \n 1.Push\n 2.Pop\n 3.Display\n 4.StackTop\n
5.StackBottom\n 6.IsEmpty\n 7.IsFull\n 8.Exit");
        printf("\nPut here : ");
        scanf("%d", &choice);
        printf("\n");
        switch (choice)
        {

        case 1:
            printf("Enter the value to push : ");
            scanf("%d", &no);
            top = push(top, no);
            break;

        case 2:
            printf("The popped item is : %d\n", pop());
            break;

        case 3:
            StackDisplay(top);
            break;

        case 4:
            printf("StackTop is : %d\n", stackTop());
            break;

        case 5:
            printf("StackBottom is : %d\n", stackBottom(top));
            break;

        case 6:
            if (isEmpty(top))
            {
                printf(" Yes Stack Empty\n");
            }
            else
            {
                printf("Not Empty\n");
            }
            break;

        case 7:
            if (isFull(top))
            {
                printf("Yes Stack Full\n");
            }
            else
            {
                printf("Not Full\n");
            }
            break;

        case 8:
            exit(0);
            break;

        default:
            break;
```

Shrihari K Mangle
CSB2-74 12010517

```
        }
    }

    return 0;
}
```

OUTPUT :

1.

```
PS E:\VIT\SECOND YEAR(SY)\SEM 2\DATA STRUCTURES(DS)\DATA STRUCT
; if ($?) { gcc stackLL.c -o stackLL } ; if ($?) { .\stackLL }
ENTER THE CHOICE :
 1.Push
 2.Pop
 3.Display
 4.StackTop
 5.StackBottom
 6.IsEmpty
 7.IsFull
 8.Exit
Put here : 1

Enter the value to push : 5
ENTER THE CHOICE :
 1.Push
 2.Pop
 3.Display
 4.StackTop
 5.StackBottom
 6.IsEmpty
 7.IsFull
 8.Exit
Put here : 1

Enter the value to push : 10
ENTER THE CHOICE :
 1.Push
 2.Pop
 3.Display
 4.StackTop
 5.StackBottom
 6.IsEmpty
 7.IsFull
 8.Exit
Put here : 1

Enter the value to push : 15
```

2.

```
Enter the value to push : 15
ENTER THE CHOICE :
 1.Push
 2.Pop
 3.Display
 4.StackTop
 5.StackBottom
 6.IsEmpty
 7.IsFull
 8.Exit
Put here : 1

Enter the value to push : 20
ENTER THE CHOICE :
 1.Push
 2.Pop
 3.Display
 4.StackTop
 5.StackBottom
 6.IsEmpty
 7.IsFull
 8.Exit
Put here : 1

Enter the value to push : 25
ENTER THE CHOICE :
 1.Push
 2.Pop
 3.Display
 4.StackTop
 5.StackBottom
 6.IsEmpty
 7.IsFull
 8.Exit
Put here : 3

25 -> 20 -> 15 -> 10 -> 5 -> NULL
```

Shrihari K Mangle
CSB2-74 12010517

```
 ENTER THE CHOICE :
  1.Push
  2.Pop
  3.Display
  4.StackTop
  5.StackBottom
  6.IsEmpty
  7.IsFull
  8.Exit
 Put here : 2

 The popped item is : 25
 ENTER THE CHOICE :
  1.Push
  2.Pop
  3.Display
  4.StackTop
  5.StackBottom
  6.IsEmpty
  7.IsFull
  8.Exit
 Put here : 2

 The popped item is : 20
 ENTER THE CHOICE :
  1.Push
  2.Pop
  3.Display
  4.StackTop
  5.StackBottom
  6.IsEmpty
  7.IsFull
  8.Exit
 Put here : 2

 The popped item is : 15
```

```
 ENTER THE CHOICE :
  1.Push
  2.Pop
  3.Display
  4.StackTop
  5.StackBottom
  6.IsEmpty
  7.IsFull
  8.Exit
 Put here : 3

 10 -> 5 -> NULL

 ENTER THE CHOICE :
  1.Push
  2.Pop
  3.Display
  4.StackTop
  5.StackBottom
  6.IsEmpty
  7.IsFull
  8.Exit
 Put here : 4

 StackTop is : 10
 ENTER THE CHOICE :
  1.Push
  2.Pop
  3.Display
  4.StackTop
  5.StackBottom
  6.IsEmpty
  7.IsFull
  8.Exit
 Put here : 5

 StackBottom is : 5
```

```
ENTER THE CHOICE :
 1.Push
 2.Pop
 3.Display
 4.StackTop
 5.StackBottom
 6.IsEmpty
 7.IsFull
 8.Exit
Put here : 6

Not Empty
ENTER THE CHOICE :
 1.Push
 2.Pop
 3.Display
 4.StackTop
 5.StackBottom
 6.IsEmpty
 7.IsFull
 8.Exit
Put here : 7

Not Full
ENTER THE CHOICE :
 1.Push
 2.Pop
 3.Display
 4.StackTop
 5.StackBottom
 6.IsEmpty
 7.IsFull
 8.Exit
Put here : 8

PS E:\VIT\SECOND YEAR(SY)\SEM 2\DATA STRUCTURES(DS)\DATA STRUCTURES LAB>
```

Shrihari K Mangle
CSB2-74 12010517

B. INFIX TO POSTFIX :

```c
#include <stdio.h>
#include <ctype.h>
#define MAX 50
char stack[50];
int top = -1;
void push(char elem)
{
    if (top == MAX - 1)
    {
        printf("\nSTACK OVERFLOW\n");
    }
    else
    {
        stack[++top] = elem;
    }
}
char pop()
{
    if (top == MAX - 1)
    {
        printf("\nSTACK OVERFLOW\n");
    }
    else
    {
        return (stack[top--]);
    }
}
int pr(char symbol)
{

    if (symbol == '^')
    {
        return (3);
    }
    else if (symbol == '*' || symbol == '/')
    {
        return (2);
    }
    else if (symbol == '+' || symbol == '-')
    {
        return (1);
    }
    else
    {
        return (0);
    }
}
int main()
```

Shrihari K Mangle
CSB2-74 12010517

```
{
    char infix[50], postfix[50], ch, elem;
    int i = 0, k = 0;
    printf("Enter Your Infix Expression : ");
    scanf("%s", infix);
    push('#');
    while ((ch = infix[i++]) != '\0')
    {
        if (ch == '(')
            push(ch);
        else if (isalnum(ch))
            postfix[k++] = ch;
        else if (ch == ')')
        {
            while (stack[top] != '(')
                postfix[k++] = pop();
            elem = pop();
        }
        else
        {
            while (pr(stack[top]) >= pr(ch))
                postfix[k++] = pop();
            push(ch);
        }
    }
    while (stack[top] != '#')
        postfix[k++] = pop();
    postfix[k] = '\0';
    printf("\nPostfix Expression : %s", postfix);
}
```

OUTPUT :

```
PS E:\VIT\SECOND YEAR(SY)\SEM 2\DATA STRUCTURES(DS)\DATA STRUCTURES LAB> cd "e:\VIT\SECOND YEAR(SY)
RUCTURES LAB\" ; if ($?) { gcc infixToPostfix.c -o infixToPostfix } ; if ($?) { .\infixToPostfix }
Enter Your Infix Expression : A*(B+C)/D-G

Postfix Expression : ABC+*D/G-
PS E:\VIT\SECOND YEAR(SY)\SEM 2\DATA STRUCTURES(DS)\DATA STRUCTURES LAB>
```

Shrihari K Mangle
CSB2-74 12010517

C. INFIX TO PREFIX

```
#define MAX 50
#include <stdio.h>
#include <string.h>
#include <ctype.h>
char s[MAX];
int top = -1;
push(char elem)
{
    if (top == MAX - 1)
    {
        printf("\nSTACK OVERFLOW\n");
    }
    else
    {
        s[++top] = elem;
    }
}
char pop()
{
    if (top == MAX - 1)
    {
        printf("\nSTACK OVERFLOW\n");
    }
    else
    {
        return (s[top--]);
    }
}
int pr(char elem)
{
    switch (elem)
    {
    case '#':
        return 0;
    case ')':
        return 1;
    case '+':
    case '-':
        return 2;
    case '*':
    case '/':
        return 3;
    }
}
int main()
{
    char infx[50], prfx[50], ch, elem;
```

```
    int i = 0, k = 0;
    printf("\n\nRead the Infix Expression ? ");
    scanf("%s", infx);
    push('#');
    strrev(infx);
    while ((ch = infx[i++]) != '\0')
    {
        if (ch == ')')
            push(ch);
        else if (isalnum(ch))
            prfx[k++] = ch;
        else if (ch == '(')
        {
            while (s[top] != ')')
                prfx[k++] = pop();
            elem = pop();
        }
        else
        {
            while (pr(s[top]) >= pr(ch))
                prfx[k++] = pop();
            push(ch);
        }
    }
    while (s[top] != '#')
        prfx[k++] = pop();
    prfx[k] = '\0';
    strrev(prfx);
    strrev(infx);
    printf("\n\nGiven Infix Expn: %s Prefix Expn: %s\n", infx, prfx);
}
```

OUTPUT :

```
Read the Infix Expression ? A*(B+C)/D-G


Given Infix Expn: A*(B+C)/D-G Prefix Expn: -*A/+BCDG
PS E:\VIT\SECOND YEAR(SY)\SEM 2\DATA STRUCTURES(DS)\DATA STRUCTURES LAB>
```