ASSIGNMENT – APPLICATION OF STACK
ARGHA MALLICK – 11500122014

Q1. Tower Of Hanoi

Source-Code:

```
#include <stdio.h>
void towerOfHanoi(int n, int start, int stop, int temp)
{
   if (n > 0)
   {
      towerOfHanoi(n - 1, start, temp, stop);
      printf("Move Disk %d from Tower %d -> %d\n", n, start, stop);
      towerOfHanoi(n - 1, temp, stop, start);
   }
}
int main()
{
   int n;
   printf("Enter the number of Disks: ");
   scanf("%d", &n);
   towerOfHanoi(n, 1, 3, 2);
   return 0;
}
```

Output:

```
linuxmint@jc198:~/Desktop/ARGHA$ gcc towerOfHanoi.c
linuxmint@jc198:~/Desktop/ARGHA$ ./a.out
Enter the number of Disks: 3
Move Disk 1 from Tower 1 -> 3
Move Disk 2 from Tower 1 -> 2
Move Disk 1 from Tower 3 -> 2
Move Disk 3 from Tower 1 -> 3
Move Disk 1 from Tower 2 -> 1
Move Disk 2 from Tower 2 -> 3
Move Disk 1 from Tower 1 -> 3
linuxmint@jc198:~/Desktop/ARGHA$
```

Q2. Reverse A String Using Stack

Source-Code:

```
#include <stdio.h>
#define MAX 100

char st[MAX];
int top = -1;

int isEmpty()
{
   if (top == -1)
      return 1;
   return 0;
}

int isFull()
{
   if (top == MAX - 1)
      return 1;
   return 0;
}

void push(char data)
{
   if (isFull() == 1)
   {
      printf("Overflow\n");
      return;
   }
   top++;
   st[top] = data;
}

char pop()
{
   if (isEmpty() == 1)
   {
      printf("Underflow\n");
      return '\0';
   }
   return st[top--];
}

int main()
{
   int n = 5;
   char str[100];
```

```
    printf("Enter the String: ");
    scanf("%s",str);
    for (int i = 0; i < n; i++)
        push(str[i]);
    for (int i = 0; i < n; i++)
        printf("%c", pop());
    printf("\n");
    return 0;
}
```

Output:

```
linuxmint@jc198:~/Desktop/ARGHA$ gcc reverse.c
linuxmint@jc198:~/Desktop/ARGHA$ ./a.out
Enter the String: ARGHA
AHGRA
linuxmint@jc198:~/Desktop/ARGHA$
```

Q3. Infix to Postfix Expression Conversion

Source-Code:

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <ctype.h>

#define MAX 100

char st[MAX];
int top = -1;

int isEmpty()
{
    return top == -1;
}

int isFull()
{
    return top == MAX - 1;
}
```

```
void push(char data)
{
   if (isFull())
   {
      printf("Overflow\n");
      return;
   }
   st[++top] = data;
}

char pop()
{
   if (isEmpty())
   {
      printf("Underflow\n");
      return '\0';
   }
   return st[top--];
}

int isOperator(char data)
{
   return (data == '+' || data == '-' || data == '*' || data == '/' || data == '^');
}

int precedence(char data)
{
   switch (data)
   {
   case '^':
      return 3;
   case '*':
   case '/':
      return 2;
   case '+':
   case '-':
      return 1;
   default:
      return 0;
   }
}

void infixToPostfix(char exp[])
{
   int n = strlen(exp);
   char output[MAX];
   int outputIndex = 0;

   push('(');

   for (int i = 0; i < n; i++)
   {
```

```
        char item = exp[i];

        if (item == '(')
            push(item);
        else if (isalpha(item))
            output[outputIndex++] = item;
        else if (isOperator(item))
        {
            while (!isEmpty() && precedence(item) <= precedence(st[top]))
            {
                output[outputIndex++] = pop();
            }
            push(item);
        }
        else if (item == ')')
        {
            while (st[top] != '(')
                output[outputIndex++] = pop();
            pop();
        }
    }

    while (!isEmpty() && st[top] != '(')
        output[outputIndex++] = pop();

    output[outputIndex] = '\0';

    printf("Postfix Expression: %s\n", output);
}

int main()
{
    char exp[100];
    printf("Enter Infix expression:\t");
    scanf("%s", exp);
    infixToPostfix(exp);
    return 0;
}
```

Output:

```
linuxmint@jc198:~/Desktop/ARGHA$ gcc infixToPostfix.c
linuxmint@jc198:~/Desktop/ARGHA$ ./a.out
Enter Infix expression: (A+B)*(C-D)
Postfix Expression: AB+CD-*
linuxmint@jc198:~/Desktop/ARGHA$
```

Q4. Evaluation of Postfix Expression

Source-Code:

```
#include <ctype.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAX 100

int st[MAX];
int top = -1;

int isEmpty() { return top == -1; }

int isFull() { return top == MAX - 1; }

void push(char data) {
  if (isFull()) {
    printf("Overflow\n");
    return;
  }
  st[++top] = data;
}

char pop() {
  if (isEmpty()) {
    printf("Underflow\n");
    return '\0';
  }
  return st[top--];
}

int evaluatePostfix(char exp[]) {
  int n = strlen(exp);

  for (int i = 0; i < n; i++) {
    char item = exp[i];

    if (isdigit(item)) {
      push(item - '0');
    } else {
      int num2 = pop();
      int num1 = pop();
      switch (item) {
      case '+':
        push(num1 + num2);
        break;
      case '-':
        push(num1 - num2);
        break;
```

```
      case '*':
        push(num1 * num2);
        break;
      case '/':
        push(num1 / num2);
        break;
      default:
        break;
      }
    }
  }
  return pop();
}

int main() {
  char exp[100]; // 553+22+/*
  printf("Enter Postfix expression: ");
  scanf("%s", exp);
  int result = evaluatePostfix(exp);
  printf("%d\n", result);
  return 0;
}
```

Output:

```
linuxmint@jc198:~/Desktop/ARGHA$ gcc evalPostfix.c
linuxmint@jc198:~/Desktop/ARGHA$ ./a.out
Enter Postfix expression: 553+22+/*
10
linuxmint@jc198:~/Desktop/ARGHA$
```