01/01/24

Data Structures

Q1. Write a program to simulate the working of stack using an array with the following:

a). PUSH
b). POP
c). Display

The program should print appropriate messages for stack outflow, stack underflow.

```
#include <stdio.h>
#include <stdlib.h>
# define N 5
int stack [N];
int top = -1;
void push()
{
  if (top == N)
  {
    printf ("Stack overflow");
  }
  else
  {
    int x;
    printf ("enter the element to be inserted");
    scanf ("%d", &x);
    top++;
    stack [top] = x;
  }
}
void pop()
{
```

```
if (top == -1)
{
    printf("Stack underflow");
}
else
{
    int y;
    y = stack[top];
    top--;
    printf("The element to be deleted is %d", y);
}
}
void display()
{
    if (top == -1)
    {
        printf("Stack is empty");
    }
    else
    {
        printf("The element in stack are: ");
        for(int i=N; i>=0; i--)
        {
            printf("%d", stack[i]);
        }
    }
}
void main()
{
    while(1)
    {
        int choice;
        printf("Enter your choice: \n 1. PUSH \n
            2. POP \n 3. DISPLAY");
```

Entered
Passed
NL
1/11/2024.

```
scanf("%d", &choice);
switch(choice)
{
    case1 :push();
    break;
    case2: pop();
    break;
    case3: display()
    break;
    case4: exit(1);
    break;
    default: printf("Invalid input.");
    break;
  }
 }
}
```

Output:- Enter the operation 1. push
2. pop
3. display
4. -1 to stop
1
Enter the number
7
Successfully pushed

```
enter the operation
 1.push
 2.pop
 3.display
 enter -1 to stop
1
enter the values
10
push operation is succesfull
1
enter the values
20
push operation is succesfull
2
20 pop() operation successfull
3
10
-1
stopping the operations

Process returned 0 (0x0)   execution time : 24.343 s
Press any key to continue.
```

Q2. Write a program to convert a given valid parenthesized infix arithmetic expression to postfix expression. The expression consists of single ~~single~~ character operands and the binary operation +, -, *, /.

```
#include <stdio.h>
#include<ctype.h>
#define size 50
char stack [size];
int top=-1;
push(char elem)
{
  stack [++top]=elem;
}
char pop()
{
  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);
  }
}
void main()
{
  char infix[50], postfix[50], ch, elem;
  int i=0, k=0;
  printf("Enter the infix expression:");
  scanf("%s", infix);
  push('#');
  while( ch = infix[i++] != '\0');
  {
    if (ch=='c')
      push(ch);
    else if (isalnum(ch)) postfix[k++] = ch;
    else if (ch == ')')
    {
      while (stack[top] != 'c')
      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("\n postfix expression = %s \n", postfix);
}
```

Output:-

(K+L-M*N+(O^P)*W/U/V*T+Q)

Infix expression: (K+L-M*N+(O^P)*W/U/V*T+Q)

Postfix expression: KL+MN*-OP^W*U/V/T*+Q+

```
Enter size of stack 3
Assume the infix expression contains single letter variables and single digit constants only.

Enter Infix expression : k*l+m(n)
Postfix Expression: kl*mn+
Process returned 0 (0x0)   execution time : 49.048 s
Press any key to continue.
```