Institute of Engineering & Management
Department of Computer Science & Engineering
Data Structure Laboratory for 2nd year 3rd semester 2017
Code: CS 392

Date: 2/8/17

ASSIGNMENT-2

Problem-1

Problem Statement: Implement Stack using array.

Algorithm:

```
Step-1: START
Step-2: Declare global variable top=-1 as integer and a integer array stack[100]
Step-3: Inside main(), declare rpt=1, i=0 as integers.
Step-4: do
            Print the commands for user
            Scan for i.
            Switch for values of i between
                case 1: call push()
                case 2: call pop()
                case 3: call display()
                default: print "wrong input".
            Ask user whether to continue or exit
            scan for rpt
        while rpt is equal to 1
Step-5: inside push(), if top is greater than or equal to 99
                print "Stack overflow"
        else scan for stack[top+1]
            top=top+1
Step-6: inside pop(), if top is less than or equal to -1
                print "Stack underflow"
        else top=top-1
Step-7: inside display(), if top is less than 0
                print "Stack empty"
        else print every element in the stack from position 0 to top
Step-8: END
```

Source code:

```
#include <stdio.h>

int top=-1;
int stack[100];

void pop();
void push();
void display();

void main()
{
      int rpt=1, i=0;
      do
      {
            printf("Choose between following operation\n '1'
                  to push operation\n '2' for pop operation\n
                              '3' to display\n");
            scanf("%d",&i);
            switch(i)
            {
                  case 1: push(); break;
                  case 2: pop(); break;
                  case 3: display(); break;
                  default : printf("Wrong input\n");
                          continue;
            }
            printf("Do u want to continue? if yes then press
                        '1' or else press any key\n");
            fflush(stdin);
            scanf("%d", &rpt);
      } while (rpt == 1);
}

void display()
{
      int i;
      if(top<0)
      {
            printf("Stack Empty\n");
            return;
      }
      printf("The elements in the stack are \n");
      for(i=0;i<=top;i++)
      {
            printf("%d, ", stack[i]);
      }
}

void push()
{
      if(top==99)
      {
            printf("Stack overflow\n");
            return;
      }
      else
      {
            printf("Enter the integer value\n");
            scanf("%d", &stack[top+1]);
            top++;
      }
}

void pop()
{
      if(top<0)
      {
            printf("Stack underflow\n");
            return;
      }
      else --top;
}
```

Input/Output:

```
Choose between following operation
 '1' to push operation
 '2' for pop operation
 ‘3' to display
1
Enter the integer value
3
Do u want to continue? if yes then press '1' or else press any key
1
Choose between following operation
 '1' to push operation
 '2' for pop operation
 ‘3' to display
1
Enter the integer value
4
Do u want to continue? if yes then press '1' or else press any key
1
Choose between following operation
 '1' to push operation
 '2' for pop operation
 ‘3' to display
3
The elements in the stack are 3,4,
Do u want to continue? if yes then press '1' or else press any key
1
Choose between following operation
 '1' to push operation
 '2' for pop operation
 ‘3' to display
2
Do u want to continue? if yes then press '1' or else press any key
1
Choose between following operation
 '1' to push operation
 '2' for pop operation
 ‘3' to display
3
The elements in the stack are 2,
Do u want to continue? if yes then press '1' or else press any key
0
```

Problem-2

Problem Statement: Convert in-fix to post-fix expression using stack

Algorithm:

```
Step-1: START
Step-2: declare global variables top=-1, optop=-1 as int & arrays postfix[100],
        infix[100], operand[100] as char.
Step-3: Inside main(), print the command for entering the in-fix expression
Step-4: take input as string in infix
Step-5: call infix_to_postfix()
Step-6: inside infix_to_postfix(), declare i and len=strlen(infix)
Step-7: initialize infix[len]=’)’ and operand[0]=’(’ & optop=0
Step-8: for i=0 to i=len+1
            if infix[i]>=’a’ and infix[i]<=’z’
                call push(infix[i])
            else if infix[i]>=’A’ and infix[i]<=’Z’
                call push(infix[i])
            else if infix[i]==’(’
                operand[optop+1]=infix[i] & optop=optop+1
            else if infix[i]==’)’
                whlie operand[optop]==’(’
                    call push(operand[optop])
                    optop=optop-1
            else if infix[i]==’^’or’*’or’/’or’+’or’-’
                while precedence of operand[optop]>= precedence of
                operand[optop]
                    call push(operand[optop])
                    optop=optop-1
                operand[optop]=infix[i] & optop=optop+1
            else print “invalid statement”
Step-9: inside push(char c), if top is greater than or equal to 99
                    print “Stack overflow”
             else postfix[top+1]=c
                top=top+1
Step-10: inside pop(), if top is less than or equal to -1
                print “Stack underflow”
                else top=top-1
Step-11: inside precedence(char c),
            switch for values of c between
            case '^': return 5;
            case '*': return 4;
            case '/': return 3;
            case '+': return 2;
            case '-': return 1;
            default:  return 0;
Step-12: END
```

Source code:

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

int top=-1, optop=-1;
char postfix[100], infix[100], operand[100];

void pop();
void push(char);
void infix_to_postfix();
int precedence(char);

void main()
{
      printf("Enter the infix expression (without any spaces
                  and <90 characters)\n");
      gets(infix);
      infix_to_postfix();
}

void infix_to_postfix()
{
      int i, len=strlen(infix);
      infix[len]=')';
      operand[0]='(';
      optop=0;
      for(i=0;i<len+1;i++)
      {
            if(infix[i]>='a' && infix[i]<='z')
                  push(infix[i]);
            else if(infix[i]>='A' && infix[i]<='Z')
                  push(infix[i]);
            else if(infix[i]=='(')
            {     operand[optop+1]='('; optop++; }
            else if(infix[i]==')')
            {
                  while(operand[optop]!='(')
                        {
                              push(operand[optop]);
                              optop--;
                        }
                        optop--;
            }
            else if(infix[i]=='^'||'*'||'/'||'+'||'-')
            {
while(precedence(operand[optop])>=precedence(infix[i]))
                        {
                              push(operand[optop]);
                              optop--;
                        }
                        operand[optop+1]=infix[i];
                        optop++;
            }
            else { printf("invalid statement\n"); return; }
      }
      printf("The postfix statement is %s\n", postfix);
      top=optop=-1;
}
```

```
void push(char c)
{
      if(top>=99)
      {
            printf("stack overflow");
            return;
      }
      else
      {
            postfix[top+1]=c;
            top++;
      }
}

void pop()
{
      if(top<0)
      {
            printf("Stack underflow\n");
            return;
      }
      else --top;
}

int precedence(char c)
{
      switch(c)
      {
            case '^':   return 5;
            case '*':   return 4;
            case '/':   return 3;
            case '+':   return 2;
            case '-':   return 1;
            default:    return 0;
      }
}
```

Input/Output:

```
Enter the infix expression (without any spaces and <90 characters)
a*(b-c)/a^b+(d+c)
The postfix statement is abc-*ab^/dc++
```