CMPE-231 DATA STRUCTURES

Laboratory Work # 4

4

Duration – 100 minutes(April 6-10) Spring 2009

This laboratory work covers Stack Data Structure(-s),Evaluation of infix expressions

✧ Experiment 1 *(PreLab task – to be prepared in the lab)*

1.1)Examine and test the below menu driven program for Insertion(I) and Deletion(D) operation using stack structure.

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#include<conio.h>
#define STACKSIZE 5
struct stack
   {
    int items[STACKSIZE];
    int top;
   } ;
 void push(struct stack *   ,int);
 int pop(struct stack *);

 void main(void)
 {
 struct stack s;
 int xx;
 char optype;
 s.top=-1;
 do
 {
  printf("ENter Operation type \n");
  scanf("\n%c",&optype);
  switch (optype)
  {
   case 'I':{
          printf("ENTER input Number \n");
          scanf("%d",&xx);
          push(&s,xx);
            break;
          }
   case 'D':{
          printf("%d %s",pop(&s)," is deleted \n");
          break;
            }
   default :printf("%s","Illegal Oparation code \n");
   }
    }while (optype!='E');
  }
```

```
int pop(struct stack *ps)
 {int x;
  if (ps->top==-1)
            {
             printf("%s","stack underflow");
             exit(1);
             }
           else{
             x=ps->items[ps->top];
             ps->top=ps->top - 1;
           }
  return x;
  }
 void push(struct stack *ps, int x)
  {
   if (ps->top == STACKSIZE-1) {
                        printf("%s","stack overflow ");
                        exit(1);
                        }
                        else{
                           ps->top=ps->top + 1;// ++(ps->top)
                           ps->items[ps->top]=x;
                           }
  }
```

1.2) Using below structure write the same menu driven Insertion/Deletion main program, Push() and Pop() functions.

```
struct  person
{
        int  empnr;
        char name[12];
        int  age;
};
struct  stack
{
        struct person  items[6];
        int top;
};
```

Use the following input data(empnr, name , age) for your menu driven program

I
123 ALI 25
I
345 AYSE 22
..
..
D
..

E

Your program will display **empnr**, **name** and **age** when you call POP() function each time.

Experiment 2 *(PreLab task – to be prepared in advance)*

Program 4.2 is prepared for eveluating a given *postfix* expression. When you run this program *postfix* expression must be given as input string and program will generate an output as numeric real number that is called as evaluated value.

Hint : postfix expression consist of one digit numeric operands and operators and only.

Example data for the program could be as follows

a) If you give input string as

9 1 4 2 / + - 2 3 * / Program will generate evaluated output as 1.0

b) If you give input string as

3 9 2 3 $ - * 6 + and evaluated value as 9.00

```
/* Program4.2  Postfix➔Evaluated value  */
#include <stdio.h>
#include<stdlib.h>
#include <math.h>
#define MAXCOLS 80

struct stack{
   int top;
   double items[MAXCOLS];
};
double eval(char[]);
double pop(struct stack *);
void push(struct stack *,double);
int empty(struct stack *);
int isdigit(char);
double oper(int,double,double);

void main(void)
{
char expr[MAXCOLS];
int pos =0;

while((expr[pos++] = getchar()) != '\n') ;

expr[--pos] = '\0';
printf("%s%s", " the original postfix expression is ",expr);
getchar();
printf("\n %f ", eval(expr));
```

```
getchar();
}

double pop(struct stack *ps)
{ double x;
  if (ps->top==-1)
                {
                 printf("%s","stack underflow");
                 exit(1);
                 }
          else{
                 x=ps->items[ps->top];
                  ps->top=ps->top - 1;
          }
  return x;
  }
  void push(struct stack *ps, double x)
  {
   if (ps->top == MAXCOLS-1) {
                                printf("%s","stack overflow ");
                                exit(1);
                                }
                             else{
                                  ps->top=ps->top + 1;
                                  ps->items[ps->top]=x;
                                  }
  }
double eval(char expr[])
{
int c , pos;
double opnd1, opnd2, value ;
struct stack s;

s.top=-1;
for(pos =0; (c=expr[position])!='\0'; pos++)
  if (isdigit(c))
       push(&s,(double)(c-'0'));
        else {
                opnd2=pop(&s);
                opnd1=pop(&s);
                value=oper(c ,opnd1 ,opnd2);
                push(&s,value);
                }
return(pop(&s));
}
```

```
int isdigit(char symb)
{
 return ( symb >= '0' && symb <= '9');
}
double oper(int symb , double  op1 , double op2)
{
  switch (symb)
{
  case '+' :return (op1 + op2);
  case '- ' :return (op1 - op2);
  case '*' :return (op1 * op2);
  case '/' :return (op1 / op2);
  case '$' :return (pow(op1 , op2));
  default : printf("%s", "illegal operation ");
                exit(1);
}
}
```