Dr. D.Y.Patil Arts, Commerce, Science College,

Pimpri, Pune-18

S.Y.B.Sc(Comp. Sci) 2024-25

Data Structures and Algorithms – I

Assignment 8: Application of Stack

Set A

a)Write a program that reverses a string of characters. The function should use a stack library (cststack.h) of stack of characters using a static implementation of the stack.

```
// reverse of string  and palindrom (cststack.h file)
#define MAX 100
struct stack
{
        int top;
        char item[MAX];
};
typedef struct stack STACK;



void initstack(STACK *s)
{
        s->top=-1;
}

int isempty(STACK *s)
{
        if (s->top==-1)
                return(1);
        else
                return(0);
}

int isfull(STACK *s)
{
        if (s->top==MAX-1)
                return(1);
        else
                return(0);
}

void push(STACK *s,char data)
{
        ++s->top;
        s->item[s->top]=data;
}

char pop(STACK *s)
{
        return(s->item[s->top--]);
}

// reverse of string  (.c file)
#include<stdio.h>
#include<string.h>
```

```
#include"cststack.h"
main()
{
	STACK s;
	char string[MAX];
	int i;
	initstack(&s);
	printf("\nInput string: ");
	gets(string);
	printf("\nOutput string:  ");

	for(i=0;string[i]!='\0';i++)
	{
		if(isfull(&s))
			printf("\nStack is full");
		else
			push(&s,string[i]);
	}
	while(!isempty(&s))
		printf("%c",pop(&s));
}
/*
[root@localhost setB]# ./a.out

Input string: hello world good morning

Output string:  gninrom doog dlrow olleh
*/
```

b)Write a program to convert an infix expressionof the form (a*(b+c)*((d-a)/b)) into its equivalent postfix notation. Consider usual precedence's of operators. Use stack library of stack of characters using static implementation.

```
#include<stdio.h>
# define MAX 50
struct STACK
{
      char stk[MAX];
      int top;
};
typedef struct STACK stack;

//initialize the stack
void initstack(stack *s)
{
      s->top=-1;
}
int isempty(stack *s)
{
      if(s->top==-1)
            return 1;
      else
            return 0;
}
int isfull(stack *s)
{
      if(s->top==MAX-1)
```

```
		return 1;
	else
		return 0;
}
void push(stack *s,char data)
{
	s->top++;
	s->stk[s->top]=data;

}
char pop(stack *s)
{
	char val;
	val=s->stk[s->top];
	s->top--;
	return(val);  //return(s->stk[s->top--]);
}
char gettop(stack *s)
{
	return(s->stk[s->top]);
}
void display(stack *s)
{
	int i;
	//   printf("\nStack Content: ");
	for(i=0;i<=s->top;i++)
		printf("%c",s->stk[i]);
}

int isoperator(char symbol)
{
	if(symbol=='+' || symbol=='-' || symbol=='*'|| symbol=='/'||
symbol=='^')
		return 1;
	else
		return 0;
}
int priority(char oper)
{
	if(oper=='^')
		return 4;
	else if(oper=='/' || oper=='*')
		return 3;
	else if(oper=='+'||oper=='-')
		return 2;
	else
		return 1;
}
int checktop(char topsymb)
{
	if(topsymb=='(' || topsymb=='{' || topsymb=='[')
		return 1;
	else
		return 0;
}
void inpostfix(char *E)
```

```
{
      int i,j=0,flag;
      char symbol,postfix[MAX],topsymb;
      stack s;
      initstack(&s);
      printf("Symbol  postfix\tstack");
      for(i=0;E[i]!='\0';i++)
      {
            symbol=E[i];
            if(symbol=='(' || symbol=='{' || symbol=='[')
                  push(&s,symbol);
            else if(isoperator(symbol)==1)
            {
                  while(priority(gettop(&s))>=priority(symbol))
                        postfix[j++]=pop(&s);
                  push(&s,symbol);

            }

            else if(symbol==')' || symbol=='}' || symbol==']')
            {
                  topsymb=pop(&s);
                  while(!checktop(topsymb))
                  {
                        postfix[j++]=topsymb;
                        topsymb=pop(&s);
                  }
            }

            else
                  postfix[j++]=symbol;
            postfix[j]='\0';
            printf("\n%c   %s\t\t",symbol,postfix);
            display(&s);


      }
      while(!isempty(&s))
      {
            postfix[j++]=pop(&s);
            postfix[j]='\0';
            printf("\n%c %s\t\t",symbol,postfix);
            display(&s);
      }

      //display(&s);
      printf("\nPostfix Expression: %s",postfix);
}
main()
{
      char expr[MAX];

      printf("\nEnter the Infix Expresion: ");
      scanf(" %s",expr);
      inpostfix(expr);
}
```

Set B

a) A postfix expression of the formab+cd-*ab/ is to be evaluated after accepting the values of a, b, c and d. The value should be accepted only once and the same value is to be used for repeated occurrence of same symbol in the expression. Formulate the problem and write a C program to solve the problem by using stack

```
#include<stdio.h>
# define MAX 30
struct STACK
{
   double stk[MAX];
   int top;
};
typedef struct STACK stack;

//initialize the stack
void initstack(stack *s)
{
   int i;
   for(i=0;i<MAX;i++)
    s->stk[i]=0;
   s->top=-1;
}
int isempty(stack *s)
{
    if(s->top==-1)
      return 1;
    else
      return 0;
}
int isfull(stack *s)
{
   if(s->top==MAX-1)
     return 1;
   else
     return 0;
}
     void push(stack *s,double data)
{
   s->top++;
   s->stk[s->top]=data;

}
double pop(stack *s)
{
   double val;
   val=s->stk[s->top];
   s->top--;
   return(val);  //return(s->stk[s->top--]);
}
void display(stack *s)
{
   int i;
   printf("\nStack: ");
   for(i=0;i<=s->top;i++)
      printf("\t%f",s->stk[i]);
}
int isdigit(char symb)
{
    if(symb>='0' && symb<='9')
      return 1;
    else
      return 0;
}
double eval(double opnd1,double opnd2,char symb)
{
int i,pow=1;
  switch(symb)
  {
    case '+': return(opnd1+opnd2);
    case '-': return(opnd1-opnd2);
    case '*': return(opnd1*opnd2);
    case '/': return(opnd1/opnd2);
    case '^': for(i=1;i<=opnd2;i++)
           pow=pow*opnd1;
           return(pow);
  }

}

int isoperator(char symbol)
{
     if(symbol=='+' || symbol=='-' || symbol=='*'|| symbol=='/'||
symbol=='^')
           return 1;
     else
           return 0;
}
char* convert(char *expr)
{
 int i,j,scnt=0,symval[10],flag=0;
  char symbol[10];                    //01234567   scnt=0,1,2,3
                     //expr=42+54-*\0
symbol[scnt++]=expr[0];         //symbol[0]=a  [1]=b  [2]=c
printf("\nEnter the Value of %c: ",expr[0]);
scanf("%d",&symval[0]);         //symval[0]=4  [1]=2  [2]=5
expr[0]=symval[0]+48;
for(i=1;expr[i]!='\0';i++)       //i=1,2,3,4             flag=1
  {
   flag=0;
   if(isoperator(expr[i])==0)
    {
      for(j=0;j<scnt;j++)        //j=0
      {
      if(expr[i]==symbol[j])
      {
           flag=1;
           expr[i]=symval[j]+48;
           break;
```

```
         }
        }
        if(flag!=1)
        {
             symbol[scnt]=expr[i];
             printf("\nEnter the Value of %c: ",expr[i]);
             scanf("%d",&symval[scnt]);
             expr[i]=symval[scnt]+48;
             scnt++;
        }

      }
    }
    printf("\n Expression after converting into digit form=%s",expr);
    return expr;
}
double posteval(char *expr)
{
  stack s;
  int i,j,scnt=0,symval[10],flag=0;
  char symbol[10];
  double opnd1,opnd2,result;
  initstack(&s);
  expr=convert(expr);
  for(i=0;expr[i]!='\0';i++)
  {

      if(isdigit(expr[i])==1)
      push(&s,expr[i]-48);
      else
      {
      opnd2=pop(&s);
      opnd1=pop(&s);
      result=eval(opnd1,opnd2,expr[i]);
      push(&s,result);
      }
  }
  return(pop(&s));
}
main()
{
    char expr[MAX];
    printf("\nEnter the Expression: ");
    scanf(" %s",expr);
    printf("\nResult after postfix Evaluation=%f",posteval(expr));
}
```

b) Write a program that checks whether a string of characters is palindrome or not. The function should use a stack library (cststack.h) of stack of characters using a static implementation of the stack.

```
//program to implement dynamic stack for palindrom
#include<stdio.h>
#include<stdlib.h>
#include"cststack.h"
main()
```

```
{
        STACK s;
        char string[MAX],str[MAX],ch;
        int i,j=0;;

        printf("\nInput string: ");
        gets(string);

        for(i=0;string[i]!='\0';i++)
        {
                if(isfull(&s))
                        printf("\nStack is full");
                else
                        push(&s,string[i]);

        }

        while(!isempty(&s))
        {
                ch=pop(&s);
                str[j++]=ch;
        }

        str[j]='\0';
        if(strcmp(string,str)==0)
                printf("\nstring is palindrom");
        else
                printf("\nstring is not palindrom");
}
/*
 [root@localhost setB]# ./a.out

Input string: madam

string is palindrom[root@localhost setB]# ./a.out

Input string: sybcs

string is not palindrom
*/
```

Set C

a) Write a program that checks the validity of parentheses in any algebraic expression. The function should use a stack library (cststack.h) of stack of characters using a static implementation of the stack.