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

Date: 9/8/17

ASSIGNMENT-2(Continued)

Problem-3

Problem Statement: Evaluate a post-fix expression.

Algorithm:

Step-1: START
Step-2: declare global variable top=-1 and a character array post[100]
Step-3: create a array of structure of character c & integer I naming value[52]
Step-4: Inside main(), print "Enter the post-fix expression" & take input of the string.
Step-5: print "The result is " & print return value of calc()
Step-6: inside calc(), call getvalue()
Step-7: inside getvalue(), declare variables i, j, count=-1 & len=strlen(post) as integer and an array of 60 caharacters
Step-8: for i=0 to i=len-1
switch for value of post[i] between
case 'A' to 'Z': count=count+1 & op[count]=post[i]
case 'A' to 'Z': count=count+1 & op[count]=post[i]
Step-9: top=top+1 & value[top].c=op[0]
Step-10: for i=0 to i=count
for j=top to j=0
if op[i] is equal to value[j].c
then break from loop.
if j is equal to -1
then top=top+1 & valu[top].c=op[i]
Step-11: for i=0 to i=top
print the character in value[i].c & scan for integer in value[i].i
Step-12: exit getvalue(), return to function calc()
Step-13: declare variables i, j, temp=-1, cal[50], len=strlen(post)
Step-14: for i=0 to i=len-1
switch for value in post[i]
case 'A' to 'Z': for j=0 to j=top
if post[i] is equal to value[j].c
then temp=temp+1 &
cal[temp]=value[j].i
case 'A' to 'Z': for j=0 to j=top
if post[i] is equal to value[j].c
then temp=temp+1 &
cal[temp]=value[j].i
case '^': cal[temp-1]=pow(cal[emp-1], cal[temp])
temp=temp-1
case '*': cal[temp-1]=temp[temp-1]*cal[temp]
temp=temp-1
case '/': cal[temp-1]=temp[temp-1]/cal[temp]
temp=temp-1

case '+': cal[temp-1]=temp[temp-1]+cal[temp]
temp=temp-1
case '-': cal[temp-1]=temp[temp-1]-cal[temp]
temp=temp-1

Step-15: return cal[0]
Step-16: END

Source code:

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

int top=-1;
char post[100];
struct charvalues
{
      char c;
      int i;
} value[52];

void getvalue();
int calc();

void main()
{
      printf("Enter the post-fix expression (without
                  spaces)\n");
      gets(post);
      printf("The result is %d\n",calc());
}

void getvalue()
{
      int len=strlen(post), i, j, count=-1;
      char op[60];
      for(i=0;i<len;i++)
            switch(post[i])
            {
                  case 'A'...'Z': op[++count]=post[i]; break;
                  case 'a'...'z': op[++count]=post[i]; break;
            }
      value[++top].c=op[0];
      for(i=0;i<=count;i++)
      {
            for(j=top;j>=0;j--)
                  if(op[i]==value[j].c)
                        break;
            if(j==-1)
                  value[++top].c=op[i];
      }
      for(i=0;i<=top;i++)
      {
            printf("\n%c = ", value[i].c);
            scanf("%d", &value[i].i);
      }
}

int calc()
```

```
{
      getvalue();
      int i, len=strlen(post), cal[50], temp=-1, j;
      for(i=0;i<=len;i++)
      {
            switch(post[i])
            {
                  case 'A'...'Z':  for(j=0;j<=top;j++)
                                   {if(post[i]==value[j].c)
                                   cal[++temp]=value[j].i;}
                                   break;
                  case 'a'...'z':  for(j=0;j<=top;j++)
                                   {if(post[i]==value[j].c)
                                   cal[++temp]=value[j].i;}
                                   break;
                  case '^':  cal[temp-1]=
                             (int)pow((double)cal[temp-1],
                                   (double)cal[temp] );
                             temp--; break;
                  case '*':  cal[temp-1]=
                                   cal[temp-1]*cal[temp];
                             temp--; break;
                  case '/':  cal[temp-1]=
                                   cal[temp-1]/cal[temp];
                             temp--; break;
                  case '+':  cal[temp-1]=
                                   cal[temp-1]+cal[temp];
                             temp--; break;
                  case '-':  cal[temp-1]=
                                   cal[temp-1]-cal[temp];
                             temp--; break;
            }
      }
      return cal[0];
}
```

Input/Output: Enter the post-fix expression (without spaces)
ab+ad^*

a = 2

b = 3

d = 5
The result is 160

Problem-4

Problem Statement: Implement Stack from both ends of a stack.

Algorithm:

Step-1: START
Step-2: declare global integer variable top1=-1, top2=100 & an array stack[100]
Step-3: in the main(), declare variables i and n=0
Step-4: do
        print the user commands for push, pop and display operation
        scan for command & store ist in i
        switch for value of i between
            case 1: call push()
            case 2: call pop()
            case 3: call display()
            default: print “wrong input”
        print “Enter 1 to continue & scan for n
    while n is equal to 1
Step-5: inside push(), declare i as integer
Step-6: check if top1 is equal to top2-1
        then print “Stack overflow”
        return
Step-7: print user commands for pushing to stack 1 or 2
Step-8: scan for i
Step-9: if i is equal to 1
        then scan for stack[top1+1]
        top1=top1+1
    else if i is equal to 2
        then scan for stack[top2-1]
        top2=top2-1
Step-10: Inside pop(), print user commands for popping from stack 1 or 2
Step-11: declare i as integer and scan for i
Step-12: if i is equal to 1
        then if top1<0
            print ”stack underflow” & return
        top1=top1-1 & print “popped”
    else if i is equal to 2
        then if top2>99
            print “stack underflow” & return
        top2=top2+1 & print “popped”
Step-13: inside display(), declare i as an integer
Step-14: if top1<0
        then print “Stack 1 is empty”
    else print “The elements in stack 1 are ”
    for i=0 to i=top1
        print stack[i]
Step-15: if top2>99
        then print “Stack 2 is empty”
    else print “The elements in the stack 2 are”
    for i=99 to i=top2
        print stack[i]

Source code:

```
#include <stdio.h>

int top1=-1, top2=100;
int stack[100];

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

void main()
{
      int i,n=0;
      do
      {
            printf("Enter the following commands\n '1' to
                              push\n '2' to pop\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");
            }
            printf("Enter 1 to continue\n");
            scanf("%d", &n);
      } while(n==1);
}

void push()
{
      if(top1==top2-1)
      {
            printf("Stack overflow"); return;
      }
      printf("Enter \n '1' for stack 1\n '2' for stack
                  2\n");
      int i;
      scanf("%d", &i);
      if(i==1)
      {
            printf("Enter the integer\n");
            top1++; scanf("%d", &stack[top1]);
      }
      else if(i==2)
      {
            printf("Enter the integer\n");
            top2--; scanf("%d", &stack[top2]);
      }
}

void pop()
{
      printf("Enter to stack no. for popping\n '1' for stack
                              1\n '2' for stack 2\n");
      int i;
      scanf("%d",&i);
```

```
	if(i==1)
	{
		if(top1<0)
		{
			printf("Stack underflow"); return;
		}
		top1--; printf("popped\n");
	}
	else if(i==2)
	{
		if(top2>99)
		{
			printf("Stack underflow"); return;
		}
		top2++; printf("popped\n");
	}
}

void display()
{
	int i;
	if(top1<0)
		printf("stack 1 is empty\n");
	else
	{
		printf("The elements in stack 1 are ");
		for(i=0;i<=top1;i++)
			printf("%d, ", stack[i]);
		printf("\n");
	}
	if(top2>99)
		printf("stack 2 is empty\n");
	else
	{
		printf("The elements in stack 2 are ");
		for(i=99;i>=top2;i--)
			printf("%d, ", stack[i]);
		printf("\n");
	}
}
```

Input/Output:

```
Enter the following commands
 '1' to push
 '2' to pop
 '3' to display
1
Enter
 '1' for stack 1
 '2' for stack 2
1
Enter the integer
32
Enter 1 to continue
1
Enter the following commands
 '1' to push
```

```
 '2' to pop
 '3' to display
1
Enter
 '1' for stack 1
 '2' for stack 2
1
Enter the integer
5
Enter the following commands
 '1' to push
 '2' to pop
 '3' to display
1
Enter
 '1' for stack 1
 '2' for stack 2
2
Enter the integer
7
Enter 1 to continue
1
Enter the following commands
 '1' to push
 '2' to pop
 '3' to display
2
Enter to stack no. for popping
 '1' for stack 1
 '2' for stack 2
2
popped
Enter 1 to continue
1
Enter the following commands
 '1' to push
 '2' to pop
 '3' to display
3
The elements in stack 1 are 32, 5,
stack 2 is empty
Enter 1 to continue
0
```

Problem-5

Problem Statement: Show steps to solve Tower of Hanoi problem using 'n' rings.

Algorithm:
Step-1: START
Step-2: Inside main(), declare n as an integer variable
Step-3: print "Enter the number of rings"
Step-4: scan for n
Step-5: call move(n, 's', 'd', 't')
Step-6: Inside move(int n, char src, char dest, char temp), if n>0
call move(n-1, src, temp, dest)
print the step i.e shift 'n' disk from 'src' to 'dest'
call move(n-1, temp, dest, src)
Step-7: END

Source code:

```
#include <stdio.h>

void move(int,char,char,char);

void main()
{
      int n;
      printf("Enter the number of rings\n");
      scanf("%d",&n);
      move(n,'s','d','t');
}

void move(int n, char src, char dest, char temp)
{
      if(n>0)
      {
            move(n-1, src, temp, dest);
            printf("shift %d disk from %c to %c\n", n, src,
                              dest);
            move(n-1, temp, dest, src);
      }
}
```

Input/Output:
Enter the number of rings
3
shift 1 disk from s to d
shift 2 disk from s to t
shift 1 disk from d to t
shift 3 disk from s to d
shift 1 disk from t to s
shift 2 disk from t to d
shift 1 disk from s to d