CSE2003- Data Structures and Algorithms

LAB ASSIGNMENT-1

Slot: L53 + L54

Faculty: GAYATHRI P Mam

Date: 26-07-20

19BCE0829 KSHEERAJ KANDRA

QUESTION-1:

1. Menu-driven C program implement stack. Perform push, pop, peek and display operations.

   PSEUDO CODE:

```
Adding into stack
procedure push(item : items);
{add item to the global stack stack;
top is the current top of stack
and n is its maximum size}
begin
if top = n then overflow;
top := top+1;
stack(top) := item;
end: {of add}
Deletion in stack
procedure pop(var item : items);
{remove top element from the stack and put it in
the item}
begin
if top = 0 then underflow;
item := stack(top);
top := top-1;
end; {of delete}
display last element:
if top=0 then stack is empty
else print stack[top]
display all elements:
```

```
for I =top ;I>0;I++
print stack[I]
```

PROGRAM CODE:

```
#include<stdio.h>
#include<stdlib.h>
#define max_size 4
int stack[max_size],top=0;
void push();
void pop();
void peep();
void display();

int main()
{
        int choice;
        do{
             printf("\nProgram to perform STACK OPERATIONS \n");
             printf("1.Push\n");
             printf("2.Pop\n");
             printf("3.Peep\n");
             printf("4.Display\n");
             printf("5.Exit\n");
             printf("\nEnter your choice: ");
             scanf("%d",&choice);
             switch(choice)
             {
                       case 1:
                                      push();
                                      break;
                       case 2:
                                      pop();
                                      break;
                       case 3:
                                      peep();
                                      break;
                       case 4:
                                      display();
```

```
                                        break;
                        case 5:
                                        exit(0);
                                        break;
                        default:
                                printf("Invalid choice:\n");
                                break;
                }
        }while(choice!=5);
return 0;
}
void push()
{
        int item;
        if(top==(max_size))
        {
                printf("\nStack Overflow\n");
        }
        else
        {
                printf("Enter the element to be inserted: ");
                scanf("%d",&item);
                top=top+1;
                stack[top]=item;
        }
}
void pop()
{
        int item;
        if(top==0)
        {
                printf("Stack Underflow");
        }
        else
        {
                item=stack[top];
                top=top-1;
                printf("\nThe poped element: %d\t",item);
        }
```

```
}
void peep()
{
        if(top==0)
        {
                printf("\nStack is empty");
        }
        else
        {
                printf("The  topmost  element  of  the  stack  is
%d",stack[top]);
        }
}
void display()
{
        int i;
        if(top==0)
        {
                printf("\nStack is Empty");
        }
        else
        {
        printf("\nThe stack elements are:\n" );
        for(i=top;i>=1;i--)
        {
                printf("%d\n",stack[i]);
        }
        }
}
```

PROGRAM OUTPUT:

19BCE0829
KANDRA KSHEERAJ

```
Program to perform STACK OPERATIONS
1.Push
2.Pop
3.Peep
4.Display
5.Exit

Enter your choice: 1
Enter the element to be inserted: 52

Program to perform STACK OPERATIONS
1.Push
2.Pop
3.Peep
4.Display
5.Exit

Enter your choice: 1
Enter the element to be inserted: 23

Program to perform STACK OPERATIONS
1.Push
2.Pop
3.Peep
4.Display
5.Exit

Enter your choice: 2

The poped element: 23
Program to perform STACK OPERATIONS
1.Push
2.Pop
3.Peep
4.Display
5.Exit

Enter your choice: 3
The topmost element of the stack is 52
Program to perform STACK OPERATIONS
1.Push
2.Pop
3.Peep
4.Display
5.Exit

Enter your choice: 4
```

```
The stack elements are:
52

Program to perform STACK OPERATIONS
1.Push
2.Pop
3.Peep
4.Display
5.Exit

Enter your choice: 5

--------------------------------
Process exited after 47.32 seconds wit
```

19BCE0829
KANDRA KSHEERAJ

Question 2:

2. C program to perform expression conversion and evaluation. Take infix expression as run-time input. Convert the expression into postfix and evaluate the postfix expression using stack.

PSEUDO CODE:

```
Infix to postfix:
while there are more symbols to be read the next
symbol case: operand --> output it.
’(’ --> push it on the stack.
’)’ --> pop operators from the stack to the output
until a ’(’ is popped; do not output either of the
parentheses.
operator --> pop higher- or equal-precedence operators
from the stack to the output; stop before popping a
lower-precedence operator or a ’(’. Push the operator
on the stack.
end case
end while
pop the remaining operators from the stack to the
output
for evaluating:
Begin
for each character ch in the postfix expression, do
if ch is an operator ? , then
a := pop first element from stack
b := pop second element from the stack
res := b ? a
push res into the stack
else if ch is an operand, then
add ch into the stack
done
return element of stack top
End
```

19BCE0829
KANDRA KSHEERAJ

PROGRAM CODE:

```
#define SIZE 50
#include <ctype.h>
#include <stdio.h>
char s[SIZE];
int top=-1;
void RemoveSpaces(char* source) {
char* i = source;
char* j = source;
     while(*j != 0) {
          *i = *j++;
          if(*i != ' ')
          i++;}
          *i = 0;}
void push(char elem) {
s[++top] = elem;
}
char pop() {
return (s[top--]);
}
int pr(char elem) {
     switch (elem) {
     case '#':
     return 0;
     case '(':
     return 1;
     case '+':
     case '-':
     return 2;
     case '*':
     case '/':
     return 3;
}
}void infix_to_postfix(char *infix, char *postfix) {
char ch, elem;
int i = 0, k = 0;
RemoveSpaces(infix);
push('#');
```

```
while ((ch = infix[i++]) != '\n') {
if (ch == '(')
push(ch);
else if (isalnum(ch))
postfix[k++] = ch;
else if (ch == ')') {
while (s[top] != '(')
postfix[k++] = pop();
elem = pop();
} else {
while (pr(s[top]) >= pr(ch))
postfix[k++] = pop();
push(ch);
}
}
while (s[top] != '#')
postfix[k++] = pop();
postfix[k] = 0;
}
int eval_postfix(char *postfix) {
char ch;
int i = 0, op1, op2;
while((ch = postfix[i++]) != 0) {
if(isdigit(ch))
push(ch-'0');
else {
op2 = pop();
op1 = pop();
     switch(ch) {
     case '+' : push(op1+op2);
     break;
     case '-' : push(op1-op2);
     break;
     case '*' : push(op1*op2);
     break;
     case '/' : push(op1/op2);
     break;
} } }
return s[top];
```

```
}
int main() {
char infx[50], pofx[50];
printf("\nInput the infix expression: ");
fgets(infx, 50, stdin);
infix_to_postfix(infx, pofx);
printf("\nGiven Infix Expression: %sPostfix Expression: %s",
infx, pofx);
top = -1;
printf("\nResult of evaluation of postfix expression: %d",
eval_postfix(pofx));
}
```

PROGRAM OUTPUT:

```
Input the infix expression: (A + B) * (C + D)

Given Infix Expression: (A+B)*(C+D)
Postfix Expression: AB+CD+*
Result of evaluation of postfix expression: 0
--------------------------------
```

```
Input the infix expression: 4*(5-(7+2))

Given Infix Expression: 4*(5-(7+2))
Postfix Expression: 4572+-*
Result of evaluation of postfix expression: -16
---------------------------------
```

QUESTION 3:

3. C program to check whether given expression is balanced expression or not using stack.

PSEUDO CODE:

```
Stack;
Loop through all characters in program {
if (symbol is an ‘open bracket’)
stack.push(symbol);
if (symbol is a ‘closed bracket’) {
if (stack.empty()) return no_match;
}
```

```
else {
}
char stacktop;
stack.top(stacktop);
stack.pop();
if (symbol and stacktop mismatch) return no_match;
     }
}
```

PROGRAM CODE:

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

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

void push(char);
void pop();
void find_top();

void main()
{
                 int i;
                 char a[100];
                 printf("Enter Expression\n");
                 scanf("%s", &a);
                 for (i = 0; a[i] != '\0';i++)
                 {
                   if (a[i] == '(')
                   {
                        push(a[i]);
                   }
                   else if (a[i] == ')')
                   {
                        pop();
                   }
                 }
```

```
                find_top();
}
void push(char a)
{
                stack[top] = a;
                top++;
}
void pop()
{
                if (top == -1)
                {
                  printf("expression is invalid\n");
                  exit(0);
                }
                else
                {
                  top--;
                }
}
void find_top()
{
                if (top == -1)
                  printf("\nexpression is valid\n");
                else
                  printf("\nexpression is invalid\n");
}
```

PROGRAM OUTPUT:

```
Enter Expression
5-(7*8))
expression is invalid

--------------------------------
```

```
Enter Expression
(5*7)*(2/5)

expression is valid

--------------------------------
```

19BCE0829
KANDRA KSHEERAJ

QUESTION 4:

4. Menu-driven C program to implement queue using array. Perform enqueue, dequeue and display operations.

PSEUDO CODE:

```
procedure enqueue(data)
if queue is full
return overflow
endif
rear ? rear + 1
queue[rear] ? data
return true
procedure dequeue
if queue is empty
return underflow
end if
data = queue[front]
front ? front + 1
return true
end procedure
display
end procedure
CODE:
if front =rear
queue is empty
else for i=0;i<rear;i++
print queue[i]
end procedure
```

PROGRAM CODE:

```
#include<stdio.h>
#include<stdlib.h>
#define max_size 4
int queue[max_size],front=-1,rear=-1;

void enqueue();
void dequeue();
```

```
void display();

 int main()
{
        int choice;
        do{

                printf("\nProgram     to     perform     QUEUE
OPERATIONS\n");
                printf("1.enqueue\n");
                printf("2.Dequeue\n");
                printf("3.Display\n");
                printf("4.Exit\n");
                printf("\nEnter your choice: ");
                scanf("%d",&choice);
                switch(choice)
                {
                        case 1:
                                        enqueue();
                                        break;
                        case 2:
                                        dequeue();
                                        break;
                        case 3:
                                        display();
                                        break;
                        case 4:
                                        exit(0);
                                        break;
                        default:
                                        printf("\nInvalid
choice:\n");
                                        break;
                }
        }while(choice!=4);
        return 0;
}
void enqueue()
{
```

```
        int item;
        if(rear==(max_size-1))
        {
                printf("\nQueue Overflow");
        }
        else
        {
                printf("Enter the element to be inserted: ");
                scanf("%d",&item);
                rear=rear+1;
                queue[rear]=item;

                if(front==-1)
                        front=0;
        }

}
void dequeue()
{
        int item;
        if(front==-1)
        {
                printf("\nQueue Underflow");
        }
        else
        {
                item=queue[front];
                printf("\nThe deleted element: %d ",item);
                if(front==rear)
                {
                        front=-1;
                        rear=-1;
                }
                else
                {
                        front=front+1;
                }
        }
}
```

```
void display()
{
        int i;
        if(front==-1)
        {
                printf("\nQueue is Empty");
        }
        else
        {
                printf("\nThe queue elements are:\n" );
                for(i=front;i<=rear;i++)
                {
                        printf("%d\t",queue[i]);
                }
        }

}
```

PROGRAM OUTPUT:

19BCE0829
KANDRA KSHEERAJ

```
Program to perform QUEUE OPERATIONS
1.enqueue
2.Dequeue
3.Display
4.Exit

Enter your choice: 1
Enter the element to be inserted: 5

Program to perform QUEUE OPERATIONS
1.enqueue
2.Dequeue
3.Display
4.Exit

Enter your choice: 1
Enter the element to be inserted: 8

Program to perform QUEUE OPERATIONS
1.enqueue
2.Dequeue
3.Display
4.Exit

Enter your choice: 2

The deleted element: 5
Program to perform QUEUE OPERATIONS
1.enqueue
2.Dequeue
3.Display
4.Exit

Enter your choice: 1
Enter the element to be inserted: 6

Program to perform QUEUE OPERATIONS
1.enqueue
2.Dequeue
3.Display
4.Exit

Enter your choice: 3
```

```
The queue elements are:
8       6
Program to perform QUEUE OPERATIONS
1.enqueue
2.Dequeue
3.Display
4.Exit

Enter your choice: 4

--------------------------------
Process exited after 29.03 seconds w
```