**Lab 5:**

1. **Write a program to implement stack using arrays.**

**Sample Input and Output:**

**1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit**

**Enter your option : 4**

**Stack is empty.**

**1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit**

**Enter your option : 2**

**Stack is underflow.**

**1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit**

**Enter your option : 3**

**Stack is empty.**

**1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit**

**Enter your option : 5**

**Stack is underflow.**

**1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit**

**Enter your option : 1**

**Enter element : 25**

**Successfully pushed.**

**1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit**

**Enter your option : 1**

**Enter element : 26**

**Successfully pushed.**

**1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit**

**Enter your option : 3**

**Elements of the stack are : 26 25**

**1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit**

**Enter your option : 2**

**Popped value = 26**

**1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit**

**Enter your option : 4**

**Stack is not empty.**

**1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit**

**Enter your option : 5**

**Peek value = 25**

**1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit**

**Enter your option : 6**

**Solution:**

**//File Name: StackUsingArray.c**

**//==================================**

**#include <stdio.h>**

**#include <stdlib.h>**

**#define STACK\_MAX\_SIZE 10**

**#include "StackOperations.c"**

**int main() {**

**int op, x;**

**while(1) {**

**printf("1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit\n");**

**printf("Enter your option : ");**

**scanf("%d", &op);**

**switch(op) {**

**case 1:**

**printf("Enter element : ");**

**scanf("%d", &x);**

**push(x);**

**break;**

**case 2:**

**pop();**

**break;**

**case 3:**

**display();**

**break;**

**case 4:**

**isEmpty();**

**break;**

**case 5:**

**peek();**

**break;**

**case 6:**

**exit(0);**

**}**

**}**

**}**

**//File Name: StackOperations.c**

**//==================================**

**int top=-1,stack[STACK\_MAX\_SIZE];**

**void push(int n)**

**{**

**if(top+1<STACK\_MAX\_SIZE)**

**{**

**top++;**

**stack[top]=n;**

**printf("Successfully pushed.\n");**

**}**

**else**

**printf("Stack is overflow.\n");**

**}**

**void pop(){**

**if(top<0)**

**{**

**printf("Stack is underflow.\n");**

**}**

**else**

**{**

**printf("Popped value = %d\n",stack[top]);**

**top--;**

**}**

**}**

**void display()**

**{**

**if(top>=0)**

**{**

**printf("Elements of the stack are : ");**

**for(int i=top;i>=0;i--)**

**{**

**printf("%d ",stack[i]);**

**}**

**}**

**else**

**{**

**printf("Stack is empty.");**

**}**

**printf("\n");**

**}**

**void isEmpty()**

**{**

**if(top<0)**

**{**

**printf("Stack is empty.\n");**

**}**

**else**

**{**

**printf("Stack is not empty.\n");**

**}**

**}**

**void peek()**

**{**

**if(top>=0)**

**printf("Peek value = %d\n",stack[top]);**

**else**

**printf("Stack is underflow.\n");**

**}**

1. **Write the code in the functions isEmpty(), push(int x), pop() and evaluatePostfix(char \*e) in the below program according to hints given as comment lines.**

**Sample Output -1**

**Enter the postfix expression : 234+-**

**Result : -5**

**Sample Output -2**

**Enter the postfix expression : 345+-34-**

**Invalid postfix expression.**

**Solutions:**

**//File Name: PostfixEvaluation.c**

**//==================================**

**#include <ctype.h>**

**#include <stdio.h>**

**#define STACK\_MAX\_SIZE 20**

**//Declare the required stack variables.**

**int stack[20];**

**int top=-1;**

**//Return 1 if stack is empty else return 0.**

**int isEmpty() {**

**}**

**//Push the character into stack**

**void push(int x) {**

**if(top>=STACK\_MAX\_SIZE-1){**

**printf("Stack overflow\n");**

**}**

**else{**

**top++;**

**stack[top]=x;**

**}**

**}**

**//pop a character from stack**

**int pop() {**

**int x;**

**if(top<0)**

**printf("Stack underflow\n");**

**else{**

**x=stack[top];**

**top=top-1;**

**return x;**

**}**

**}**

**//Output Format - Result : <result> if the input postfix expression is vaild.**

**//Output Format - Invalid postfix expression,. - if the input expression is invalid.**

**//postfix expression is given as the parameter.**

**void evaluatePostfix(char \* e) {**

**int num,n1,n2,n3,c1=0,c2=0;**

**while(\*e!='\0')**

**{**

**if(isdigit(\*e))**

**{**

**num=\*e-48;**

**push(num);**

**c1++;**

**}**

**else if(\*e=='+'||\*e=='-'||\*e=='\*'||\*e=='/')**

**{**

**n1=pop();**

**n2=pop();**

**switch(\*e)**

**{**

**case '+':**

**{**

**n3=n1+n2;**

**break;**

**}**

**case '-':**

**{**

**n3=n2-n1;**

**break;**

**}**

**case '\*':**

**{**

**n3=n1\*n2;**

**break;**

**}**

**case '/':**

**{**

**n3=n2/n1;**

**break;**

**}**

**}**

**push(n3);**

**c2++;**

**}**

**else**

**{**

**printf("Invalid postfix expression.\n");**

**break;**

**}**

**e++;**

**}**

**if(c1-c2==1)**

**printf("Result : %d\n",pop());**

**else**

**printf("Invalid postfix expression.\n");**

**}**

**//Read a postfix expression and evaluate it.**

**int main() {**

**char exp[20];**

**char \*e, x;**

**printf("Enter the postfix expression : ");**

**scanf("%s",exp);**

**e = exp;**

**evaluatePostfix(e);**

**}**

**Home Assignment**

1. **Write a C program to convert a Prefix expression to Postfix expression.**

**Solution :**

**//File Name: prefixToPostfix.c**

**//==================================**

**#include<stdio.h>**

**#include<string.h>**

**#include<stdlib.h>**

**#define MAX 20**

**char str[MAX],stack[MAX];**

**int top=-1;**

**void push(char c)**

**{**

**stack[++top]=c;**

**}**

**char pop()**

**{**

**return stack[top--];**

**}**

**void pre\_post()**

**{**

**int n,i,j=0;**

**char c[20];**

**char a,b,op;**

**printf("Enter a Prefix expression:");**

**gets(str);**

**n=strlen(str);**

**for(i=0;i<MAX;i++)**

**{**

**stack[i]='\0';**

**}**

**printf("Postfix expression is:");**

**for(i=0;i<n;i++)**

**{**

**if(str[i]=='+'||str[i]=='-'||str[i]=='\*'||str[i]=='/')**

**{**

**push(str[i]);**

**}**

**else**

**{**

**c[j++]=str[i];**

**while((top!=-1)&&(stack[top]=='@'))**

**{**

**a=pop();**

**c[j++]=pop();**

**}**

**push('@');**

**}**

**}**

**c[j]='\0';**

**printf("%s\n",c);**

**}**

**void main()**

**{**

**pre\_post();**

**}**

1. **Write a C program to convert an Infix expression to Prefix expression.**

**Solution:**

**//File Name: infixToPrefix.c**

**//==================================**

**#include<stdio.h>**

**#include<string.h>**

**#include<stdlib.h>**

**#include<conio.h>**

**#define max 100**

**int top=-1, a[max];**

**void push(char x)**

**{**

**a[++top]=x;**

**}**

**char pop(){**

**if(top==-1)**

**return -1;**

**else**

**return a[top--];**

**}**

**int prcd(char c)**

**{**

**if(c==')')**

**return 0;**

**else if(c=='+'|| c=='-')**

**return 1;**

**else if(c=='\*'|| c=='/')**

**return 2;**

**}**

**void strrev(char \*exp)**

**{**

**int size=strlen(exp);**

**int j=size,i=0;**

**char temp[size];**

**temp[j--]='\0';**

**while(exp[i]!='\0')**

**{**

**temp[j]=exp[i];**

**j--;**

**i++;**

**}**

**strcpy(exp,temp);**

**}**

**void infixtoprefix(char infix[max],char prefix[max])**

**{**

**char temp,x;**

**int i=0,j=0;**

**strrev(infix);**

**while(infix[i]!='\0')**

**{**

**temp=infix[i];**

**if(isalnum(temp))**

**{**

**prefix[j++]=temp;**

**}**

**else if(temp==')')**

**push(temp);**

**else if(temp=='(')**

**{**

**while((x=pop())!=')')**

**{**

**prefix[j++]=x;**

**}**

**}**

**else**

**{**

**while(prcd(a[top])>=prcd(temp))**

**{**

**prefix[j++]=pop();**

**}**

**push(temp);**

**}**

**i++;**

**}**

**while(top!=-1)**

**prefix[j++]=pop();**

**prefix[j]='\0';**

**strrev(prefix);**

**}**

**void main()**

**{**

**char infix[max],prefix[max];**

**printf("Enter Infix Expression:");**

**gets(infix);**

**infixtoprefix(infix,prefix);**

**printf("Prefix Expression:%s\n",prefix);**

**}**

1. **Write the code in the functions convertInfix(char \*e), priority(char x), push(char x), pop() and isEmpty in the below code according to the hints given as comment lines.**
2. **Sample Output -1**
3. **Enter the expression : A+B\*(C+D)**
4. **Postfix expression : ABCD+\*+**
5. **Sample Output -2**
6. **Enter the expression : A+D\*C+E-F&D**
7. **Invalid symbols in infix expression. Only alphanumeric and { '+', '-','\*', '%', '/' } are allowed.**
8. **Sample Output -3**
9. **Enter the expression : A+B\*C+(D\*E**
10. **Invalid infix expression : unbalanced parenthesis.**

**Solution:**

**//File Name: Infix2PostfixMain.c**

**//==================================**

**#include "Infix2PostfixOperation.c"**

**int main() {**

**char exp[20];**

**char \*e, x;**

**printf("Enter the expression : ");**

**scanf("%s",exp);**

**e = exp;**

**convertInfix(e);**

**}**

**//File Name: Infix2PostfixOperation.c**

**//==================================**

**#include <stdlib.h>**

**#include <string.h>**

**#include <stdio.h>**

**#include <ctype.h>**

**#define STACK\_MAX\_SIZE 20**

**char stack [STACK\_MAX\_SIZE];**

**int top = -1;**

**//Push the character into stack**

**void push(char x) {**

**stack[++top]=x;**

**}**

**//pop a character from stack**

**char pop() {**

**if(top==-1)**

**return -1;**

**else**

**return stack[top--];**

**}**

**// Return 0 if char is '('**

**// Return 1 if char is '+' or '-'**

**// Return 2 if char is '\*' or '/' or '%'**

**int priority(char x) {**

**if(x=='(')**

**return 0;**

**if(x=='+'||x=='-')**

**return 1;**

**if(x=='\*'||x=='/'||x=='%')**

**return 2;**

**return 0;**

**}**

**//Output Format**

**//if expression is correct then output will be Postfix Expression : <postfix notation>**

**//If expression contains invalid operators then output will be "Invalid symbols in infix expression. Only alphanumberic and { '+', '-','\*', '%%', '/' } are allowed."**

**//If the expression contains unbalanced paranthesis the output will be "Invalid infix expression : unbalanced parenthesis."**

**void convertInfix(char \* e) {**

**char x;**

**char \*p;**

**p=e;**

**int c1=0,c2=0,c3=0,c4=1;**

**while(\*p!='\0')**

**{**

**if(isalnum(\*p)||\*p=='+'||\*p=='-'||\*p=='\*'||\*p=='/'||\*p=='%')**

**c1++;**

**else if(\*p=='(')**

**c2++;**

**else if(\*p==')')**

**c3++;**

**else**

**{**

**c4++;**

**}**

**p++;**

**}**

**if(c4!=1){**

**printf("Invalid symbols in infix expression. Only alphanumeric and { '+', '-', '\*', '%', '/' }\n");**

**}**

**else if(c2!=c3)**

**printf("Invalid infix expression : unbalanced parenthesis.\n");**

**else**

**{**

**printf("Postfix expression : ");**

**while(\*e!='\0')**

**{**

**if(isalnum(\*e))**

**printf("%c",\*e);**

**else if(\*e=='(')**

**{**

**push(\*e);**

**}**

**else if(\*e==')')**

**{**

**while((x=pop())!='(')**

**printf("%c",x);**

**}**

**else**

**{**

**while(priority(stack[top])>=priority(\*e))**

**printf("%c",pop());**

**push(\*e);**

**}**

**e++;**

**}**

**while(top!=-1)**

**{**

**printf("%c",pop());**

**}**

**printf("\n");**

**}**

**}**

**4. Write a program to implement stack using linked lists.**

**Sample Input and Output:**

**1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit**

**Enter your option : 1**

**Enter element : 33**

**Successfully pushed.**

**1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit**

**Enter your option : 1**

**Enter element : 22**

**Successfully pushed.**

**1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit**

**Enter your option : 1**

**Enter element : 55**

**Successfully pushed.**

**1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit**

**Enter your option : 1**

**Enter element : 66**

**Successfully pushed.**

**1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit**

**Enter your option : 3**

**Elements of the stack are : 66 55 22 33**

**1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit**

**Enter your option : 2**

**Popped value = 66**

**1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit**

**Enter your option : 2**

**Popped value = 55**

**1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit**

**Enter your option : 3**

**Elements of the stack are : 22 33**

**1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit**

**Enter your option : 5**

**Peek value = 22**

**1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit**

**Enter your option : 4**

**Stack is not empty.**

**1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit**

**Enter your option : 6**

**Solution :**

**//File Name: StackUsingLL.c**

**//==================================**

**#include <stdio.h>**

**#include <stdlib.h>**

**#include "StackOperationsLL.c"**

**int main() {**

**int op, x;**

**while(1) {**

**printf("1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit\n");**

**printf("Enter your option : ");**

**scanf("%d", &op);**

**switch(op) {**

**case 1:**

**printf("Enter element : ");**

**scanf("%d", &x);**

**push(x);**

**break;**

**case 2:**

**pop();**

**break;**

**case 3:**

**display();**

**break;**

**case 4:**

**isEmpty();**

**break;**

**case 5:**

**peek();**

**break;**

**case 6:**

**exit(0);**

**}**

**}**

**}**

**//File Name: StackOperationsLL.c**

**//==================================**

**struct node**

**{**

**int data;**

**struct node\*next;**

**}\*top=NULL;**

**typedef struct node\* N;**

**void push(int value){**

**N newNode;**

**newNode=(N)malloc(sizeof(struct node));**

**newNode->data=value;**

**if(top==NULL)**

**{**

**newNode->next=NULL;**

**}**

**else**

**{**

**newNode->next=top;**

**}**

**top=newNode;**

**printf("Successfully pushed.\n");**

**}**

**void pop()**

**{**

**if(top==NULL)**

**{**

**printf("Stack is underflow.\n");**

**}**

**else**

**{**

**N temp=top;**

**printf("Popped value = %d\n",temp->data);**

**top=temp->next;**

**free(temp);**

**}**

**}**

**void display()**

**{**

**if(top==NULL)**

**{**

**printf("Stack is empty.\n");**

**}**

**else**

**{**

**N temp;**

**temp=top;**

**printf("Elements of the stack are : ");**

**do{**

**printf("%d ",temp->data);**

**temp=temp->next;**

**}**

**while(temp!=NULL);**

**printf("\n");**

**}**

**}**

**void isEmpty()**

**{**

**if(top==NULL)**

**printf("Stack is empty.\n");**

**else**

**printf("Stack is not empty.\n");**

**}**

**void peek()**

**{**

**if(top!=NULL)**

**{**

**N temp=top;**

**printf("Peek value = %d\n",temp->data);**

**}**

**else**

**{**

**printf("Stack is underflow.\n");**

**}**

**}**

**5. Write a C program to convert a Postfix expression to Prefix expression.**

**Solution:**

**//File Name: postfixToPrefix.c**

**//==================================**

**#include<stdio.h>**

**#include<string.h>**

**#include<stdlib.h>**

**#define MAX 20**

**char str[MAX],stack[MAX];**

**int top=-1;**

**void push(char c)**

**{**

**stack[++top]=c;**

**}**

**char pop()**

**{**

**return stack[top--];**

**}**

**int check(char ch)**

**{**

**return (ch>='a' && ch<='z')||(ch>='A' && ch<='Z');**

**}**

**int isOperator(char x)**

**{**

**switch(x){**

**case '+':**

**case '-':**

**case '/':**

**case '\*':**

**return 1;**

**}**

**return 0;**

**}**

**void postfixToprefix()**

**{**

**int n,i,j=0;**

**char c[20];**

**char a,b,op;**

**printf("Enter the postfix expression:");**

**scanf("%s",str);**

**n=strlen(str);**

**for(i=0;i<MAX;i++)**

**stack[i]='\0';**

**printf("Prefix expression is:");**

**for(i=n-1;i>=0;i--)**

**{**

**if(isOperator(str[i]))**

**{**

**push(str[i]);**

**}**

**else**

**{**

**c[j++]=str[i];**

**while((top!=-1) && (stack[top]=='#'))**

**{**

**a=pop();**

**c[j++]=pop();**

**}**

**push('#');**

**}**

**}**

**c[j]='\0';**

**i=0;**

**j=strlen(c)-1;**

**char d[20];**

**while(c[i]!='\0'){**

**d[j--]=c[i++];**

**}**

**printf("%s\n",d);**

**}**

**int main()**

**{**

**postfixToprefix();**

**return 0;**

**}**

**LAB 6 QUEUE**

**LAB WORK**

1. **Write a program to implement queue using arrays.**

**Sample Input and Output:**

**1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit**

**Enter your option : 1**

**Enter element : 23**

**Successfully inserted.**

**1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit**

**Enter your option : 1**

**Enter element : 56**

**Successfully inserted.**

**1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit**

**Enter your option : 3**

**Elements in the queue : 23 56**

**1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit**

**Enter your option : 4**

**Queue is not empty.**

**1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit**

**Enter your option : 5**

**Queue size : 2**

**1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit**

**Enter your option : 2**

**Deleted element = 23**

**1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit**

**Enter your option : 2**

**Deleted element = 56**

**1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit**

**Enter your option : 4**

**Queue is empty.**

**1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit**

**Enter your option : 6**

**Solution:**

**//File Name: QueueUsingArray.c**

**//==================================**

**#include <conio.h>**

**#include <stdio.h>**

**#include "QueueOperations.c"**

**int main() {**

**int op, x;**

**while(1) {**

**printf("1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit\n");**

**printf("Enter your option : ");**

**scanf("%d",&op);**

**switch(op) {**

**case 1:**

**printf("Enter element : ");**

**scanf("%d",&x);**

**enqueue(x);**

**break;**

**case 2:**

**dequeue();**

**break;**

**case 3:**

**display();**

**break;**

**case 4:**

**isEmpty();**

**break;**

**case 5:**

**size();**

**break;**

**case 6: exit(0);**

**}**

**}**

**}**

**//File Name: QueueOperations.c**

**//==================================**

**#include<stdio.h>**

**#include<stdlib.h>**

**#define MAX 20**

**int queue\_arr[MAX];**

**int rear=-1;**

**int front=-1;**

**void enqueue(int item)**

**{**

**if(isFull())**

**{**

**printf("Queue overflow\n");**

**}**

**else**

**{**

**if(front==-1)**

**front=0;**

**rear=rear+1;**

**queue\_arr[rear]=item;**

**printf("Successfully inserted.\n");**

**}**

**}**

**void dequeue()**

**{**

**int item;**

**if(front==-1 || front==rear+1)**

**{**

**printf("Queue is underflow.\n");**

**}**

**else{**

**item=queue\_arr[front];**

**front=front+1;**

**printf("Deleted element = %d\n",item);**

**}**

**}**

**void isEmpty()**

**{**

**if(front==-1 || front==rear+1)**

**printf("Queue is empty.\n");**

**else**

**printf("Queue is not empty.\n");**

**}**

**int isFull()**

**{**

**if(rear==MAX-1)**

**return 1;**

**else**

**return 0;**

**}**

**void display()**

**{**

**int i;**

**if(front==-1 || front==rear+1)**

**{**

**printf("Queue is empty.\n");**

**}**

**else**

**{**

**printf("Elements in the queue : ");**

**for(i=front;i<=rear;i++)**

**printf("%d ",queue\_arr[i]);**

**printf("\n");**

**}**

**}**

**void size()**

**{**

**int i,count=0;**

**if(front==-1 || front==rear+1)**

**printf("Queue size : 0\n");**

**else**

**{**

**for(i=front;i<=rear;i++)**

**count++;**

**printf("Queue size : %d\n",count);**

**}**

**}**

1. **Write a program to implement circular queue using linked lists.**

**Sample Input and Output:**

**1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit**

**Enter your option : 1**

**Enter element : 15**

**Successfully inserted.**

**1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit**

**Enter your option : 1**

**Enter element : 16**

**Successfully inserted.**

**1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit**

**Enter your option : 1**

**Enter element : 17**

**Successfully inserted.**

**1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit**

**Enter your option : 3**

**Elements in the circular queue : 15 16 17**

**1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit**

**Enter your option : 5**

**Circular queue size : 3**

**1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit**

**Enter your option : 2**

**Deleted value = 15**

**1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit**

**Enter your option : 2**

**Deleted value = 16**

**1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit**

**Enter your option : 2**

**Deleted value = 17**

**1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit**

**Enter your option : 3**

**Circular queue is empty.**

**1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit**

**Enter your option : 4**

**Circular queue is empty.**

**1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit**

**Enter your option : 5**

**Circular queue size : 0**

**1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit**

**Enter your option : 6**

**Solution:**

**//File Name: CQueueLL.c**

**//==================================**

**#include <stdlib.h>**

**#include <stdio.h>**

**#include "CQueueOperationsLL.c"**

**int main() {**

**int op, x;**

**while(1) {**

**printf("1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit\n");**

**printf("Enter your option : ");**

**scanf("%d",&op);**

**switch(op) {**

**case 1:**

**printf("Enter element : ");**

**scanf("%d",&x);**

**enqueue(x);**

**break;**

**case 2:**

**dequeue();**

**break;**

**case 3:**

**display();**

**break;**

**case 4:**

**isEmpty();**

**break;**

**case 5:**

**size();**

**break;**

**case 6: exit(0);**

**}**

**}**

**}**

**//File Name: CQueueOperationsLL.c**

**//==================================**

**struct node**

**{**

**int data;**

**struct node\*next;**

**};**

**struct node\*f=NULL;**

**struct node\*r=NULL;**

**void enqueue(int d)**

**{**

**struct node\*n;**

**n=(struct node\*)malloc(sizeof(struct node));**

**n->data=d;**

**n->next=NULL;**

**if((r==NULL) && (f==NULL))**

**{**

**f=r=n;**

**r->next=f;**

**}**

**else**

**{**

**r->next=n;**

**r=n;**

**n->next=f;**

**}**

**printf("Successfully inserted.\n");**

**}**

**void dequeue()**

**{**

**struct node\*t;**

**t=f;**

**if((f==NULL)&&(r==NULL))**

**{**

**printf("Circular queue is underflow.\n");**

**}**

**else if(f==r)**

**{**

**f=r=NULL;**

**printf("Deleted value = %d\n",t->data);**

**free(t);**

**}**

**else**

**{**

**f=f->next;**

**r->next=f;**

**printf("Deleted value = %d\n",t->data);**

**free(t);**

**}**

**}**

**void display()**

**{**

**struct node\*t;**

**t=f;**

**if((f==NULL)&&(r==NULL))**

**{**

**printf("Circular queue is empty.\n");**

**}**

**else**

**{**

**printf("Elements in the circular queue : ");**

**do{**

**printf("%d ",t->data);**

**t=t->next;**

**}while(t!=f);**

**printf("\n");**

**}**

**}**

**void isEmpty()**

**{**

**if((f==NULL)&&(r==NULL))**

**printf("Circular queue is empty.\n");**

**else**

**printf("Circular queue is not empty.\n");**

**}**

**void size()**

**{**

**int count=0;**

**struct node\*t;**

**t=f;**

**if((f==NULL)&&(r==NULL))**

**printf("Circular queue size : 0\n");**

**else**

**{**

**do{**

**t=t->next;**

**count++;**

**}while(t!=f);**

**printf("Circular queue size : %d\n",count);**

**}**

**}**

**Home Assignment**

1. **Implementation of double ended queue using linked list  
     
   Fill the missing code int functions inject (int ele), eject, and display() in the below code.  
     
   The function inject(int ele) inserts an element ele into the rear of double ended queue and it gives "Dequeue is overflow." error if the maximum capacity of the dequeue is reached.  
     
   The function eject() removes an element from the rear of the double ended queue and it gives "Dequeue is underflow." error if there are no more elements in the dequeue.  
     
   The function display() prints all the elements of the double ended queue.**
2. **Write a program to implement circular queue using arrays.**

**Note: Define the MAX value as 5.**

**Solution**

**//File Name: CQueueUsingArray.c**

**//==================================**

**#include <stdio.h>**

**#include <stdlib.h>**

**#include "CQueueOperations.c"**

**int main() {**

**int op, x;**

**while(1) {**

**printf("1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit\n");**

**printf("Enter your option : ");**

**scanf("%d",&op);**

**switch(op) {**

**case 1:**

**printf("Enter element : ");**

**scanf("%d",&x);**

**enqueue(x);**

**break;**

**case 2:**

**dequeue();**

**break;**

**case 3:**

**display();**

**break;**

**case 4:**

**isEmpty();**

**break;**

**case 5:**

**size();**

**break;**

**case 6: exit(0);**

**}**

**}**

**}**

**//File Name: CQueueOperations.c**

**//==================================**

**#include<stdio.h>**

**#include<stdlib.h>**

**#define MAX 5**

**int cqueue\_arr[MAX];**

**int front=-1;**

**int rear=-1;**

**void enqueue(int item)**

**{**

**if(isFull())**

**printf("Circular queue is overflow.\n");**

**else**

**{**

**if(front==-1)**

**front=0;**

**if(rear==MAX-1)**

**rear=0;**

**else**

**rear=rear+1;**

**cqueue\_arr[rear]=item;**

**printf("Successfully inserted.\n");**

**}**

**}**

**void dequeue()**

**{**

**int item;**

**if(front==-1)**

**{**

**printf("Circular queue is underflow.\n");**

**}**

**else**

**{**

**item=cqueue\_arr[front];**

**if(front==rear)**

**{**

**front=-1;**

**rear=-1;**

**}**

**else if(front==MAX-1)**

**front=0;**

**else**

**front=front+1;**

**printf("Deleted element = %d\n",item);**

**}**

**}**

**void display()**

**{**

**int i;**

**if(front==-1)**

**{**

**printf("Circular queue is empty.\n");**

**}**

**else**

**{**

**printf("Elements in the circular queue : ");**

**i=front;**

**if(front<=rear)**

**{**

**while(i<=rear)**

**printf("%d ",cqueue\_arr[i++]);**

**}**

**else**

**{**

**while(i<=MAX-1)**

**printf("%d ",cqueue\_arr[i++]);**

**i=0;**

**while(i<=rear)**

**printf("%d ",cqueue\_arr[i++]);**

**}**

**printf("\n");**

**}**

**}**

**void isEmpty()**

**{**

**if(front==-1)**

**printf("Circular queue is empty.\n");**

**else**

**printf("Circular queue is not empty.\n");**

**}**

**int isFull()**

**{**

**if((front==0 && rear==MAX-1)||(front==rear+1))**

**return 1;**

**else**

**return 0;**

**}**

**void size()**

**{**

**int i,count=0;**

**if(front==-1)**

**printf("Circular queue size : 0\n");**

**else**

**{**

**i=front;**

**if(front<=rear)**

**{**

**while(i<=rear)**

**{**

**count++;**

**i++;**

**}**

**}**

**else**

**{**

**while(i<=MAX-1)**

**{**

**count++;**

**i++;**

**}**

**i=0;**

**while(i<=rear)**

**{**

**i++;**

**count++;**

**}**

**}**

**printf("Circular queue size : %d\n",count);**

**}**

**}**

1. **Implementation of double ended queue using array  
     
   Fill the missing code int functions inject (int ele), eject, and display() in the below code.  
     
   The function inject(int ele) inserts an element ele into the rear of double ended queue and it gives "Double ended queue is overflow." error if the maximum capacity of the dequeue is reached.  
     
   The function eject() removes an element from the rear of the double ended queue and it gives "Double ended queue is underflow." error if there are no more elements in the dequeue.  
     
   The function display() prints all the elements of the double ended queue.  
     
   Note: Do use printf() with '\n' at the end of display().**

**Solution:**

**//File Name: DequeueArrayMain1.c**

**//==================================**

**#include <stdio.h>**

**#include <conio.h>**

**#include "DequeueArrayInjectEject.c"**

**int main() {**

**int op, x;**

**while (1) {**

**printf("1.Inject 2.Eject 3.Display 4.Exit\n");**

**printf("Enter your option : ");**

**scanf("%d", & op);**

**switch (op) {**

**case 1:**

**printf("Enter element : ");**

**scanf("%d", & x);**

**inject(x);**

**break;**

**case 2:**

**eject();**

**break;**

**case 3:**

**display();**

**break;**

**case 4:**

**exit(0);**

**}**

**}**

**}**

**//File Name: DequeueArrayInjectEject.c**

**//==================================**

**#define MAX 10**

**int deQueue[MAX];**

**int rear =-1,front=-1;**

**void inject(int x) {**

**if(isFull())**

**{**

**printf("Double ended queue is overflow.\n");**

**}**

**else {**

**if(front==-1)**

**{**

**front=0;**

**rear=0;**

**}**

**else if(rear==MAX-1)**

**rear=0;**

**else**

**rear=rear+1;**

**deQueue[rear]=x;**

**printf("Successfully inserted at rear side.\n");**

**}**

**}**

**void eject() {**

**int item;**

**if(isEmpty())**

**{**

**printf("Double ended queue is undeflow.\n");**

**}**

**else**

**{**

**item=deQueue[rear];**

**if(front==rear)**

**{**

**front=-1;**

**rear=-1;**

**}**

**else if(rear==0)**

**rear=MAX-1;**

**else**

**rear=rear-1;**

**printf("Deleted element from the rear side = %d\n",item);**

**}**

**}**

**int isFull()**

**{**

**if((front==0 && rear==MAX-1)||(front==rear+1))**

**return 1;**

**else**

**return 0;**

**}**

**int isEmpty()**

**{**

**if(front==-1)**

**return 1;**

**else**

**return 0;**

**}**

**void display() {**

**int i;**

**if(isEmpty())**

**{**

**printf("Double ended queue is empty.\n");**

**}**

**else**

**{**

**printf("Elements in the double ended queue : ");**

**i=front;**

**if(front<=rear)**

**{**

**while(i<=rear)**

**printf("%d ",deQueue[i++]);**

**}**

**else**

**{**

**while(i<=MAX-1)**

**printf("%d ",deQueue[i++]);**

**i=0;**

**while(i<=rear)**

**printf("%d ",deQueue[i++]);**

**}**

**printf("\n");**

**}**

**}**

1. **Write a program to implement queue using linked lists.**

**Sample Input and Output:**

**1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit**

**Enter your option : 1**

**Enter element : 57**

**Successfully inserted.**

**1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit**

**Enter your option : 1**

**Enter element : 87**

**Successfully inserted.**

**1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit**

**Enter your option : 5**

**Queue size : 2**

**1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit**

**Enter your option : 3**

**Elements in the queue : 57 87**

**1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit**

**Enter your option : 2**

**Deleted value = 57**

**1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit**

**Enter your option : 2**

**Deleted value = 87**

**1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit**

**Enter your option : 3**

**Queue is empty.**

**1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit**

**Enter your option : 5**

**Queue size : 0**

**1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit**

**Enter your option : 6**

**Solution:**

**//File Name: QueueUsingLL.c**

**//==================================**

**#include <conio.h>**

**#include <stdio.h>**

**#include "QueueOperationsLL.c"**

**int main() {**

**int op, x;**

**while(1) {**

**printf("1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit\n");**

**printf("Enter your option : ");**

**scanf("%d",&op);**

**switch(op) {**

**case 1:**

**printf("Enter element : ");**

**scanf("%d",&x);**

**enqueue(x);**

**break;**

**case 2:**

**dequeue();**

**break;**

**case 3:**

**display();**

**break;**

**case 4:**

**isEmpty();**

**break;**

**case 5:**

**size();**

**break;**

**case 6: exit(0);**

**}**

**}**

**}**

**//File Name: QueueOperationsLL.c**

**//==================================**

**struct node{**

**int info;**

**struct node\*link;**

**}\*front=NULL,\*rear=NULL;**

**void enqueue(int item)**

**{**

**struct node\*tmp;**

**tmp=(struct node\*)malloc(sizeof(struct node));**

**if(tmp==NULL)**

**{**

**printf("Memory not available\n");**

**}**

**else**

**{**

**tmp->info=item;**

**tmp->link=NULL;**

**if(front==NULL)**

**front=tmp;**

**else**

**rear->link=tmp;**

**rear=tmp;**

**printf("Successfully inserted.\n");**

**}**

**}**

**void dequeue()**

**{**

**struct node\*tmp;**

**int item;**

**if(front==NULL)**

**{**

**printf("Queue is underflow.\n");**

**}**

**else**

**{**

**tmp=front;**

**item=tmp->info;**

**front=front->link;**

**printf("Deleted value = %d\n",item);**

**free(tmp);**

**}**

**}**

**void display()**

**{**

**struct node\*ptr;**

**ptr=front;**

**if(front==NULL)**

**{**

**printf("Queue is empty.\n");**

**}**

**else**

**{**

**printf("Elements in the queue : ");**

**while(ptr!=NULL)**

**{**

**printf("%d ",ptr->info);**

**ptr=ptr->link;**

**}**

**printf("\n");**

**}**

**}**

**void isEmpty()**

**{**

**if(front==NULL)**

**{**

**printf("Queue is empty.\n");**

**}**

**else**

**{**

**printf("Queue is not empty.\n");**

**}**

**}**

**void size()**

**{**

**int count=0;**

**struct node\*p;**

**p=front;**

**if(front==NULL)**

**printf("Queue size : 0\n");**

**else**

**{**

**while(p!=NULL)**

**{**

**p=p->link;**

**count++;**

**}**

**printf("Queue size : %d\n",count);**

**}**

**}**