**Ex: 1 Dt:**

**Aim:**

Create a structure student {roll no, name, blood group, mobile no, message}. Make a function that sends message to all the students who are of a particular blood group.

**Program:**

#include<stdio.h>

#include<conio.h>

struct student

{ char roll\_no[11];

char name[20];

char bloodgroup[10];

char mobileno[11];

char message[50];

};

void send(struct student \*);

static int f;

void main()

{ struct student s[3],c[3];

int i,j=1; char a[20];

printf("REGISTRATION \n");

for(i=1;i<=3;i++)

{ printf("%d \n",i);

printf("NAME \t"); gets(s[i].name);

printf("ROLL\_NO \t"); gets(s[i].roll\_no);

printf("BLOOD GROUP \t"); gets(s[i].bloodgroup);

printf("MOBILE\_NOS \t"); gets(s[i].mobileno);

}

printf("BLOODGROUP TO BE FIND \t"); gets(a);

for(i=1;i<=3;i++)

{ if(strcmp(s[i].bloodgroup,a)==0)

{ c[j]=s[i]; j++; f++; }

}

if(f==0)

{ printf("\n NOT FOUND !!"); }

else

{ printf("\n %d STUDENT WITH SAME BLOODGROUP",f);

send(c);

}

getch();

}

void send(struct student \*c)

{ int i; char m[20];

printf("\n GIVE MESSAGE \t"); gets(m);

for(i=1;i<=f;i++)

{ printf("\n TO \t"); puts(c[i].name);

strcpy(c[i].message,m);

printf("MESSAGE \t"); puts(c[i].message);

}

}

**Sample Input-Output:**

1

NAME yuvraj

ROLL\_NO 123

BLOOD GROUP a

MOBILE\_NOS 12345

2

NAME xyz

ROLL\_NO 122

BLOOD GROUP b

MOBILE\_NOS 12345

3

NAME abc

ROLL\_NO 121

BLOOD GROUP c

MOBILE\_NOS 12345

BLOODGROUP TO BE FIND a

1 STUDENT WITH SAME BLOODGROUP

GIVE MESSAGE NEED

TO yuvraj

MESSAGE NEED

**Ex: 2 Dt:**

**Aim:**

Create a flight booking structure {date, from, to, cost}. Make a function to book the ticket. Show the cost of the ticket the same when the ticket is booked a month before. Otherwise show a cost multiplied by the percentage closer to the date of travel.

**Program:**

#include<stdio.h>

#include<conio.h>

struct booking

{ int dd,mm,yy;

char from[20];

char to[20];

float cost;

};

void main()

{ struct booking b[3];

int i,d,m,y,f=0;

for(i=1;i<=3;i++)

{printf("\n");

printf("BOOKING\_NO:%d \n",i);

printf("FROM \t"); scanf("%s",&b[i].from);

printf("TO \t"); scanf("%s",&b[i].to);

printf("DATE OF FLIGHT \t"); scanf("%d%d%d",&b[i].dd,&b[i].mm,&b[i].yy);

printf("CURRENT DATE \t"); scanf("%d%d%d",&d,&m,&y);

b[i].cost=6000;

if(b[i].yy>y)

{ while(m!=12)

{ m++; f++; }

if(m==12)

{ m=1; f++;

while(b[i].mm>m)

{ m++; f++; }

}

if(f>=2)

{ b[i].cost=b[i].cost-1000; printf("COST:%.2f",b[i].cost); }

else

{ printf("COST:%.2f",b[i].cost); }

}

else

{ if(b[i].mm<m || b[i].yy<y || (b[i].mm==m && b[i].yy==y && b[i].dd<d))

{ printf("INVALID ENTRY"); break; }

if(b[i].mm>=m+2)

{ b[i].cost=b[i].cost-1000; printf("COST:%.2f",b[i].cost); }

else

{ printf("COST:%.2f",b[i].cost); }

}

}

getch();

}

**Sample Input-Output:**

BOOKING NO : 1

FROM ABC

TO XYZ

DATE OF FLIGHT 02 08 2017

CURRENT DATE 22 08 2016

INVALID ENTRY

BOOKING NO : 2

FROM UDAIPUR

TO JAIPUR

DATE OF FLIGHT 02 02 2000

CURRENT DATE 02 11 1999

COST : 5000.00

BOOKING NO : 3

FROM DABOK

TO MUMBAI

DATE OF FLIGHT 02 08 2017

CURRENT DATE 01 08 2017

COST: 6000.00

**Ex: 3 Dt:**

**Aim:**

Create a structure for RTU exam\_form {rollno, name, theory\_subjects, labs, total\_cost}.

Fill the form by taking the names of subjects and calculate the total exam fee.

**Program:**

#include<stdio.h>

#include<conio.h>

struct student

{

char roll\_no[11];

char name[20];

char sub[3][20];

char labs[3][20];

int cost;

};

void form(struct student \*);

void display(struct student \*);

void main()

{

struct student s[3];

int i,j;

char sb[3][20];

char lb[3][20];

printf("REGISTRATION \n");

for(i=0;i<3;i++)

{

printf("NAME \t"); gets(s[i].name);

printf("ROLL-no \t"); gets(s[i].roll\_no);

s[i].cost=0;

}

printf("THEORY SUBJECT NAME \n");

for(i=0;i<3;i++)

{

gets(sb[i]);

}

printf("LABS SUBJECT NAME \n");

for(i=0;i<3;i++)

{

gets(lb[i]);

}

for(i=0;i<3;i++)

{

for(j=0;j<3;j++)

{

strcpy(s[i].sub[j],sb[j]);

strcpy(s[i].labs[j],lb[j]);

}

}

form(s);

display(s);

getch();

}

void form(struct student\*s)

{

int i,j,sbcost,lbcost;

printf("ONE THEORY\_SUBJECT FORM FEES \t"); scanf("%d",&sbcost);

printf("ONE LABS SUBJECT FORM FEES \t"); scanf("%d",&lbcost);

for(i=0;i<3;i++)

{

for(j=0;j<3;j++)

{

s[i].cost=s[i].cost+sbcost+lbcost;

}

}

}

void display(struct student \*s)

{

int i;

printf("DISPLAY \n");

for(i=0;i<3;i++)

{

printf("\n NAME \t"); puts(s[i].name);

printf("ROLL-no \t"); puts(s[i].roll\_no);

printf("TOTAL EXAM FORM FEES %d",s[i].cost);

}

}

**Sample Input-Output:**

REGISTRATION

NAME YUVRAJ

ROLL-NO 123

NAME ABC

ROLL-NO 122

NAME XYZ

ROLL-NO 121

THEORY SUBJECT NAME

DSA

LINUX

OOPS

LAB SUBJECT NAME

EDC LB

DEL B

C++LB

ONE THEORY SUBJECT FORM FEES 100

ONE LAB SUBJECT FORM FEES 100

DISPLAY

NAME YUVRAJ

ROLL-NO 123

TOTAL EXAM FORM FEES 600

NAME ABC

ROLL-NO 122

TOTAL EXAM FORM FEES 600

NAME XYZ

ROLL-NO 121

TOTAL EXAM FORM FEES 600

**Ex: 4 Dt:**

**Aim:**

Create a items structure {item\_name, price, availability} & shopping cart structure {name,

phone\_no, items\_purchased, total}. List the items to the customer and allow them to

purchase the items if they are available and calculate the total bill amount.

**Program:**

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

#include<string.h>

struct shop

{

char item\_name[20];

int price;

int a;

};

struct customer

{

char name[20];

char phone\_no[11];

char cart[10][20];

};

void main()

{

struct shop s[3];

struct customer c[3];

int i,q,f=0,ch,n,x,y=1;

char u[20],p[20];

printf("ITEMS\_NAME \n");

for(i=1;i<=3;i++)

{

printf("NAME \t"); scanf("%s",&s[i].item\_name);

printf("PRICE \t"); scanf("%d",&s[i].price);

printf("QUANTITY \t"); scanf("%d",&s[i].a);

}

printf("REGISTER CUSTOMER DETAIL \n");

for(i=1;i<=3;i++)

{

printf("NAME \t"); scanf("%s",&c[i].name);

printf("PHONE\_NO \t"); scanf("%s",&c[i].phone\_no);

}

system("CLS");

printf("USERNAME \t"); scanf("%s",&u);

printf("PHONE\_NO \t"); scanf("%s",&p);

system("CLS");

for(i=1;i<=3;i++)

{

if((strcmp(c[i].name,u)==0) && (strcmp(c[i].phone\_no,p)==0))

{

f=1; q=i; break;

}

}

l: if(f==1)

{

for(i=1;i<=3;i++)

{

printf("%d %s",i,s[i].item\_name); printf("\n");

}

scanf("%d",&ch);

switch(ch)

{

case 1: printf("ITEM\_NAME %s",s[ch].item\_name); printf("\n");

printf("PRICE %d",s[ch].price); printf("\n");

printf("QUANTITY %d",s[ch].a); printf("\n");

if(s[ch].a>=1)

{

printf("1. BUY NOW 2. BUY LATER \n"); scanf("%d",&n);

if(n==1)

{

strcpy(c[q].cart[y],s[ch].item\_name);

s[ch].a=s[ch].a-1;

y++;

}

}

else

{

printf("out of stock !!"); }

break;

case 2: printf("ITEM\_NAME %s",s[ch].item\_name); printf("\n");

printf("PRICE %d",s[ch].price); printf("\n");

printf("QUANTITY %d",s[ch].a); printf("\n");

if(s[ch].a>=1)

{

printf("1. BUY NOW 2. BUY LATER \n"); scanf("%d",&n);

if(n==1)

{

strcpy(c[q].cart[y],s[ch].item\_name);

s[ch].a=s[ch].a-1;

y++;

}

}

else

{

printf("out of stock !!"); }

break;

case 3: printf("ITEM\_NAME %s",s[ch].item\_name); printf("\n");

printf("PRICE %d",s[ch].price); printf("\n");

printf("QUANTITY %d",s[ch].a); printf("\n");

if(s[ch].a>=1)

{

printf("1. BUY NOW 2. BUY LATER \n"); scanf("%d",&n);

if(n==1)

{

strcpy(c[q].cart[y],s[ch].item\_name);

s[ch].a=s[ch].a-1;

y++;

}

}

else

{

printf("out of stock !!");

}

break;

default: printf("invalid choice");

}

printf("\n");

printf("1. BUY MORE \n 2. SEE YOUR ACCOUNT \n 3.EXIT");

scanf("%d",&x);

system("CLS");

switch(x)

{

case 1: goto l;

break;

case 2: printf("\n NAME %s \t",c[q].name);

printf("\n PHONE\_NO %s\t",c[q].phone\_no);

printf("CART \n");

for(i=1;i<=y;i++)

{

printf("%s",c[q].cart[i]); printf("\n"); }

break;

case 3: exit(0);

}

}

else

{

printf("invlid username and phone\_no"); }

getch();

}

**Sample Input-Output:**

ITEMS\_NAME

NAME PENCIL

PRICE 3

QUANTITY 5

NAME RUBBER

PRICE 3

QUANTITY 5

NAME SHARPNER

PRICE 3

QUANTITY 0

REGISTER CUSTOMER DETAIL

NAME YUVRAJ

PHONE\_NO 123

NAME ABC

PHONE\_NO 122

NAME XYZ

PHONE\_NO 121

USERNAME YUVRAJ

PHONE\_NO 123

1. PENCIL
2. RUBBER
3. SHARPNER

ITEM\_NAME PENCIL

PRICE 3

QUANTITY 5

1. BUY NOW 2. BUY LATER 1
2. BUY MORE
3. SEE YOUR ACCOUNT
4. EXIT 1
5. PENCIL
6. RUBBER
7. SHARPNER

ITEM\_NAME SHARPNER

PRICE 3

QUANTITY 0

OUT OF STOCK !!

1. BUY MORE
2. SEE YOUR ACCOUNT
3. EXIT 2

NAME YUVRAJ

PHONE\_NO 123

CART

PENCIL

**Ex: 6 Dt:**

**Aim:**

Write a program to represent a 4-variable polynomial using array. Use this representation to implement addition of two input polynomials.

**Program:**

#include<stdio.h>

#include<conio.h>

void main()

{ int p1[3][5],p2[3][5],p3[6][5],i,j,i1,i2,j1,j2,k=0,s=0,si,flag;

printf("\nEnter 1st polynomial\n");

for(i=0;i<3;i++)

{ printf("\nEnter %d Term Constant and Powers of A,B,C and D: ",i);

scanf("%d%d%d%d%d",&p1[i][0],&p1[i][1],&p1[i][2],&p1[i][3],&p1[i][4]);

}printf("\nEnter 2nd polynomial\n");

for(i=0;i<3;i++)

{ printf("\nEnter %d Term Constant and Powers of A,B,C and D: ",i);

scanf("%d%d%d%d%d",&p2[i][0],&p2[i][1],&p2[i][2],&p2[i][3],&p2[i][4]);

}

for(i1=0;i1<3;i1++)

{ s=0;flag=0;

for(i2=0;i2<3;i2++)

{ si=0;

for(j1=1;j1<=4;j1++)

{ if(p1[i1][j1]!=p2[i2][j1])

{ si++; break; }

}

if(si==0)

{ p3[k][0]=p1[i1][0]+p2[i2][0];

p3[k][1]=p1[i1][1];

p3[k][2]=p1[i1][2];

p3[k][3]=p1[i1][3];

p3[k][4]=p1[i1][4];

k++; flag=100;

}

else

{ s++;}

}

if(s!=0&&flag!=100)

{ p3[k][0]=p1[i1][0];

p3[k][1]=p1[i1][1];

p3[k][2]=p1[i1][2];

p3[k][3]=p1[i1][3];

p3[k][4]=p1[i1][4];

k++;

}

}

for(i2=0;i2<3;i2++)

{ s=0;

for(i1=0;i1<3;i1++)

{ for(j1=1;j1<=4;j1++)

{ if(p2[i2][j1]!=p1[i1][j1])

{ s++;break; }

}

}

if(s==3)

{ p3[k][0]=p2[i2][0];

p3[k][1]=p2[i2][1];

p3[k][2]=p2[i2][2];

p3[k][3]=p2[i2][3];

p3[k][4]=p2[i2][4];

k++;

}

}

printf("\n---Polynomial is ---\n\n");

for(i=0;i<k;i++)

{ printf("%d",p3[i][0]);

if(p3[i][1]!=0)

{ printf("a^%d ",p3[i][1]); }

if(p3[i][2]!=0)

{ printf("b^%d ",p3[i][2]); }

if(p3[i][3]!=0)

{ printf("c^%d ",p3[i][3]); }

if(p3[i][4]!=0)

{ printf("d^%d ",p3[i][4]); }

if(i!=(k-1))

{ printf(" + ");}

}

getch();

}

**Sample Input-Output:**

ENTER 1ST POLYNOMIAL

ENTER 0 TERM CONSTANT AND POWER OF A,B,C AND D: 3 3 2 1 1

ENTER 1 TERM CONSTANT AND POWER OF A,B,C AND D: 6 1 1 2 0

ENTER 2 TERM CONSTANT AND POWER OF A,B,C AND D: 9 1 1 1 1

ENTER 2ND POLYNOMIAL

ENTER 0 TERM CONSTANT AND POWER OF A,B,C AND D: 10 3 1 1 1

ENTER 1 TERM CONSTANT AND POWER OF A,B,C AND D: 4 3 2 1 1

ENTER 2 TERM CONSTANT AND POWER OF A,B,C AND D: 6 1 1 1 0

------POLYNOMIAL IS ---

7a^3 b^2 c^1 d^1 + 6a^1 b^1 c^2 +9a^1 b^1 c^1 d^1 +10a^3 b^1 c^1 d^1 + 6a^1 b^1 c^1

**Ex: 7 Dt:**

**Aim:**

Write a C program to implement the student stack {name, age, address, phone} and perform Push and Pop operations.

**Program:**

#include<stdio.h>

#include<conio.h>

#include<string.h>

#define max 5

struct student

{ char name[20];

int age;

char address[20];

char phone[11];

}s[max];

void main()

{ int i,ch,top=-1;

while(1)

{

printf("\n 1 PUSH \n 2 POP \n 3 PEEK \n 4 TRAVERSE \n 5 EXIT \n");

scanf("%d",&ch);

switch(ch)

{

case 1: if(top==max-1)

{

printf("STACK IS OVERFLOW");

}

else

{

top++; printf("\n");

printf("NAME \t"); scanf("%s",&s[top].name);

printf("AGE \t"); scanf("%d",&s[top].age);

printf("ADDRESS \t"); scanf("%s",&s[top].address);

printf("PHONE\_NO \t"); scanf("%s",&s[top].phone);

}

break;

case 2: if(top==-1)

{

printf("STACK IS EMPTY");

}

else

{

printf("POP ELEMENT IS \n");

printf("\n NAME \t %s",s[top].name);

printf("AGE \t %d",s[top].age);

printf("ADDRESS \t %s",s[top].address);

printf("PHONE\_NO \t %s",s[top].phone);

top--;

}

break;

case 3: if(top==-1)

{

printf("STACK IS EMPTY \n");

}

else

{

printf("PEEK ELEMENT IS \n");

printf("\n NAME \t %s",s[top].name);

printf("AGE \t %d",s[top].age);

printf("ADDRESS \t %s",s[top].address);

printf("PHONE\_NO \t %s",s[top].phone);

}

break;

case 4: if(top==-1)

{

printf("STACK IS EMPTY \n"); }

else

{

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

{

printf("TRAVERSE \n");

printf("\n NAME \t %s",s[i].name);

printf("\n AGE \t %d",s[i].age);

printf("\n ADDRESS \t %s",s[i].address);

printf("\n PHONE\_NO \t %s",s[i].phone);

}

}

break;

case 5: exit(0);

default: printf("WRONG CHOICE \n");

}

}

getch();

}

**Sample Input-Output:**

1. PUSH
2. POP
3. PEEK
4. TRAVERSE
5. EXIT 1

NAME YUVRAJ AGE 18

ADDRESS AKOLA PHONE\_NO 12345

1. PUSH
2. POP
3. PEEK
4. TRAVERSE
5. EXIT 1

NAME ABC AGE 12

ADDRESS ABC PHONE\_NO 12345

1. PUSH
2. POP
3. PEEK
4. TRAVERSE
5. EXIT 1

NAME XYZ AGE 13

ADDRESS XYZ PHONE\_NO 12345

1. PUSH
2. POP
3. PEEK
4. TRAVERSE
5. EXIT 1

STACK IS OVERFLOW

1. PUSH
2. POP
3. PEEK
4. TRAVERSE
5. EXIT 2

POP ELEMENT IS

NAME XYZ AGE 13 ADDRESS XYZ PHONE\_NO 12345

1. PUSH

1. POP
2. PEEK
3. TRAVERSE
4. EXIT 4

NAME YUVRAJ

AGE 18

ADDRESS AKOLA

PHONE\_NO 12345

NAME ABC

AGE 12

ADDRESS ABC

PHONE\_NO 12345

1. PUSH

1. POP
2. PEEK
3. TRAVERSE
4. EXIT 5

**Ex: 8 Dt:**

**Aim:**

Write a C program to implement the job queue {file\_name, dt\_created, size, number\_of\_pages} and perform enqueue and dequeue operations.

**Program:**

#include<stdio.h>

#include<conio.h>

#include<process.h>

#define max 10

struct job

{ char file\_name[20];

int dt\_created,size,number\_of\_pages;

}s[max];

void main()

{ int i,ch,front=-1,rear=-1;

while(1)

{ printf("\n 1 ENQUEUE \n 2 DEQUEUE \n 3 TRAVERSE \n 4 EXIT \n");

scanf("%d",&ch);

switch(ch)

{ case 1: if(rear==max-1)

{ printf("QUEUE IS FULL"); }

else

{ rear++;

printf("\n FILE\_NAME \t"); scanf("%s",&s[rear].file\_name);

printf("DATE\_CREATED \t"); scanf("%d",&s[rear].dt\_created);

printf("SIZE \t");

scanf("%d",&s[rear].size);

printf("NUMBER\_OF\_PAGES \t"); scanf("%d",&s[rear].number\_of\_pages);

}

break;

case 2: if(rear==front)

{ printf("QUEUE IS EMPTY"); }

else

{ front++;

printf("DEQUEUE ELEMENTS \n");

printf("\n FILE\_NAME \t %s",s[front].file\_name);

printf("DATE\_CREATED \t %d",s[front].dt\_created);

printf("SIZE \t %d",s[front].size);

printf("NUMBER\_OF\_PAGES \t%d",s[front].number\_of\_pages); }

break;

case 3: if(rear==front)

{ printf("QUEUE IS EMPTY"); }

else

{for(i=front+1;i<=rear;i++)

{ printf("\n-------------------------------\n");

printf("\n FILE\_NAME \t %s",s[i].file\_name);

printf("\nDATE\_CREATED \t %d",s[i].dt\_created);

printf("\nSIZE \t %d",s[i].size);

printf("\nNUMBER\_OF\_PAGES \t ,%d",s[i].number\_of\_pages);

}

}

break;

case 4: exit(0);

default: printf("wrong choice");

}

}

getch();

}

**Sample Input-Output:**

1. ENQUEUE
2. DEQUEUE
3. TRAVERSE
4. EXIT 1

FILE\_NAME ABC

DATE\_CREATED 10

SIZE 10

NUMBER\_OF\_PAGES 25

1. ENQUEUE
2. DEQUEUE
3. TRAVERSE
4. EXIT 1

FILE\_NAME XYZ

DATE\_CREATED 21

SIZE 20

NUMBER\_OF\_PAGES 32

1. ENQUEUE
2. DEQUEUE
3. TRAVERSE
4. EXIT 1

FILE\_NAME ESC

DATE\_CREATED 12

SIZE 10

NUMBER\_OF\_PAGES 25

1. ENQUEUE
2. DEQUEUE
3. TRAVERSE
4. EXIT 2

DEQUEUE ELEMENT

FILE\_NAME ABC DATE\_CREATED 10 SIZE 10

1. ENQUEUE

1. DEQUEUE
2. TRAVERSE
3. EXIT 3

------------------------------------------------------------------------------------------------

FILE\_NAME XYZ

DATE\_CREATED 21

SIZE 20

NUMBER\_OF\_PAGES 32

FILE\_NAME ESC

DATE\_CREATED 12

SIZE 10

NUMBER\_OF\_PAGES 25

--------------------------------------------------------------------------------------------

1. ENQUEUE

1. DEQUEUE
2. TRAVERSE
3. EXIT

**Ex: 9 Dt:**

**Aim:**

Create a priority queue of patients {name, age, problem, token\_no}. Perform enqueue and dequeue operations based on the priority of token. Also, count for the number of patients with specific problem.

**Program:**

#include<stdio.h>

#include<conio.h>

#define max 5

struct patients

{ char name[20],prbl[20];

int age,tkn,ct;

}p[max],t;

int front=-1,rear=-1;

void enqueue()

{ int i;

if(rear==max-1)

{ printf("ENTRY FULL !!!"); }

else

{ rear++;

printf("ENTER TOKEN\_NO: "); scanf("%d",&p[rear].tkn);

printf("NAME \t"); scanf("%s",&p[rear].name);

printf("AGE \t"); scanf("%d",&p[rear].age);

printf("PROBLEM \t"); scanf("%s",&p[rear].prbl);

}

}

void dequeue()

{ if(front==rear)

{ printf("NO PATIENTS THERE !!!"); }

else

{ front++;

printf("\nNAME: %s",p[front].name);

printf("\nAGE: %d",p[front].age);

printf("\nPROBLEM: %s",p[front].prbl);

}

}

void traverse()

{ int i;

if(front==rear)

{ printf("NO PATIENTS THERE !!!"); }

else

{ for(i=front+1;i<=rear;i++)

{ printf("\nTOKEN\_NO: %d",p[i].tkn);

printf("\nNAME: %s",p[i].name);

printf("\nAGE: %d",p[i].age);

printf("\nPROBLEM: %s",p[i].prbl);

printf("\nPATIENTS WITH SAME PROBLEM: %d",p[i].ct);

}

}

}

void priority()

{ int i,j;

if(rear>=0)

{ for(i=front+1;i<=rear;i++)

{ for(j=i+1;j<=rear;j++)

{ if(p[i].tkn>p[j].tkn)

{ t=p[i]; p[i]=p[j]; p[j]=t; }

}

}

}

}

void check()

{ int i,j,s=0;

if(rear>0)

{ for(i=front+1;i<=rear;i++)

{ for(j=i+1;j<=rear;j++)

{ if(strcmp(p[i].prbl,p[j].prbl)==0)

{ s++; }

}p[i].ct=s; s=0;

}

}

}

void main()

{ int data,ch;

while(1)

{ system("CLS");

printf("\n1.enqueue\n2.dequeue\n3.traverse\n4.exit\n");

printf("ENTER YOUR CHOICE \t");

scanf("%d",&ch);

priority();

switch(ch)

{ case 1:enqueue();

break;

case 2:dequeue();

break;

case 3:check();

traverse();

break;

case 4:exit(0);

default:printf("INVALID CHOICE");

}

getch();

}

getch();

}

**Sample Input-Output:**

1. **ENQUEUE**
2. **DEQUEUE**
3. **TRAVERSE**
4. **EXIT**

**ENTER YOUR CHOICE 1**

**ENTER TOKEN\_NO: 1**

**NAME abc  
AGE 12  
PROBLEM abc**

**-----------------------------------------**

**ENTER YOUR CHOICE 1**

**ENTER TOKEN\_NO: 2**

**NAME xyz  
AGE 12  
PROBLEM xyz**

**------------------------------------------**

**ENTER YOUR CHOICE 1**

**ENTER TOKEN\_NO: 3**

**NAME esc  
AGE 12  
PROBLEM abc**

**-----------------------------------------**

**ENTER YOUR CHOICE 1**

**ENTER TOKEN\_NO: 4**

**NAME qx  
AGE 12  
PROBLEM xyz**

**-------------------------------------------**

**ENTER YOUR CHOICE 1**

**ENTER TOKEN\_NO: 5**

**NAME ec  
AGE 12  
PROBLEM abc**

**--------------------------------------------**

**ENTER YOUR CHOICE 3**

**ENTER TOKEN\_NO: 1**

**NAME abc  
AGE 12  
PROBLEM abc**

**PATIENTS WITH SAME PROBLEM 2**

**ENTER TOKEN\_NO: 2**

**NAME xyz  
AGE 12  
PROBLEM xyz**

**PATIENTS WITH SAME PROBLEM 1**

**ENTER TOKEN\_NO: 3**

**NAME esc  
AGE 12  
PROBLEM abc**

**PATIENTS WITH SAME PROBLEM 1**

**ENTER TOKEN\_NO: 4**

**NAME qx  
AGE 12  
PROBLEM xyz**

**PATIENTS WITH SAME PROBLEM 0**

**ENTER TOKEN\_NO: 5**

**NAME ec  
AGE 12  
PROBLEM abc**

**PATIENTS WITH SAME PROBLEM 0**

**----------------------------------------------------------**

**Ex: 10 Dt:**

**Aim:**

Write a program in C to convert and Infix expression to Postfix form. Also, display the

intermediate steps.

**Program:**

#include<stdio.h>

#include<conio.h>

char stack[20],step[20];

int top = -1,z=0;

void push(char x)

{ stack[++top] = x;

step[z]=stack[top];

z++;

}

char pop()

{ if(top == -1)

{ return(-1); }

else

{ return(stack[top--]); }

}

char stp()

{ if(z == -1)

{ return(-1); }

else

{ return(step[z--]); }

}

int priority(char x)

{ if(x == '(')

{ return(0); }

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

{ return(1); }

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

{ return(2); }

}

void main()

{

char exp[20],\*e, x;

printf("Enter the expression :: "); scanf("%s",exp);

e = exp;

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 STACK ELEMENT \n");

while(z!=-1)

{ printf("%c",stp()); }

getch();

}

**Sample Input-Output:**

Enter the expression :: (a+b\*c/d)

The result of expression :: abc\*d/+

STACK ELEMENT

/\*+(

**Ex: 11 Dt:**

**Aim:**

Write a program to evaluate a given Postfix expression and display the result.

**Program:**

#include<stdio.h>

#include<conio.h>

int stack[20],top = -1;

void push(int x)

{ stack[++top] = x; }

int pop()

{ return stack[top--]; }

void main()

{ char exp[20],\*e;

int n1,n2,n3,num;

printf("Enter the expression :: "); scanf("%s",exp);

e = exp;

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

{ if(isdigit(\*e))

{ num = \*e - 48;

push(num);

}

else

{ 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);

}

e++;

}

printf("\nThe result of expression %s = %d\n\n",exp,pop());

getch();

}

**Sample Input-Output:**

Enter the expression :: 245+\* The result of expression 245+\*=18

**Ex: 12 Dt:**

**Aim:**

Create a singly linked list of books {name, author names (2D array), pages, price}and

perform inserting elements at the beginning, end and at a specific position in the list.

**Program:**

#include<stdio.h>

#include<conio.h>

void main()

{ struct node

{ struct node \*next;

char n[20];

char a\_n[5][20];

int pg;

int pr;

}\*start=NULL,\*t,\*temp;

int ch,i,page,price,pos,c;

char name[20],a\_name[5][20];

while(1)

{l: system("CLS");

printf("1.INSERT\n2.DISPLAY\n3.EXIT");

scanf("%d",&ch);

switch(ch)

{ case 1: t=(struct node \*)malloc(sizeof(struct node));

printf("BOOK\_NAME: "); scanf("%s",&name);

printf("AUTHOR\_NAME \n");

for(i=0;i<5;i++)

{ scanf("%s",&a\_name[i]); }

printf("PAGES: "); scanf("%d",&page);

printf("PRICE: "); scanf("%d",&price);

strcpy(t->n,name);

for(i=0;i<5;i++)

{ strcpy(t->a\_n[i],a\_name[i]); }

t->pg=page;

t->pr=price;

printf("\n 1.AT FRONT \n 2.AT POS \n 3.AT LAST"); scanf("%d",&ch);

switch(ch)

{ case 1: if(start==NULL)

{ start=t; t->next=NULL; }

else

{ t->next=start; start=t; }

goto l;

case 2: if(start==NULL)

{ start=t; t->next=NULL; }

else

{ printf("enter pos"); scanf("%d",&pos);

c=1; temp=start;

while(c<pos-1)

{ temp=temp->next; c++; } t->next=temp->next; temp->next=t;

}

goto l;

case 3: if(start==NULL)

{ start=t; t->next=NULL; }

else

{ temp=start;

while(temp->next!=NULL)

{ temp=temp->next; } temp->next=t; t->next=NULL;

}goto l;

default: printf("invlaid entry"); goto l;

}

case 2: if(start==NULL)

{ printf("NO DATA THERE!!"); }

else

{ temp=start; system("CLS");

while(temp!=NULL)

{ printf("BOOK\_NAME: %s",temp->n);

printf("\nAUTHOR\_NAME");

for(i=0;i<5;i++)

{ printf("\n%s",temp->a\_n[i]); }

printf("\nPAGES: %d",temp->pg);

printf("\nPRICE: %d",temp->pr);

printf("\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");

temp=temp->next;

}

}break;

case 3: exit(0);

default: printf("invalid");

}getch();

}

}

**Sample Input-Output:**

1. **INSERT**
2. **DISPLAY**
3. **EXIT 1**

**BOOK\_NAME: EDC**

**AUTHOR\_NAME**

**XYZ ESC CSE ABC QWERTY**

**PAGES: 100 PRICE: 100**

1. **AT FRONT**
2. **AT POS**
3. **AT LAST 1**
4. **INSERT**
5. **DISPLAY**
6. **EXIT 1**

**BOOK\_NAME: DSA**

**AUTHOR\_NAME**

**XYZ ESC CSE ABC QWERTY**

**PAGES: 100 PRICE: 100**

1. **AT FRONT**
2. **AT POS**
3. **AT LAST 3**
4. **INSERT**
5. **DISPLAY**
6. **EXIT 1**

**BOOK\_NAME: OOPS**

**AUTHOR\_NAME**

**XYZ ESC CSE ABC QWERTY**

**PAGES: 100 PRICE: 100**

1. **AT FRONT**
2. **AT POS**
3. **AT LAST 2**

**ENTER POS 2**

1. **INSERT**
2. **DISPLAY**
3. **EXIT 2**

**BOOK\_NAME: EDC**

**AUTHOR\_NAME:**

**XYZ**

**ESC**

**CSE**

**ABC**

**QWERTY**

**PAGES: 100 PRICE: 100**

**BOOK\_NAME: OOPS**

**AUTHOR\_NAME:**

**XYZ**

**ESC**

**CSE**

**ABC**

**QWERTY**

**PAGES: 100 PRICE: 100**

**BOOK\_NAME: DSA**

**AUTHOR\_NAME:**

**XYZ ESC CSE ABC QWERTY**

**PAGES: 100 PRICE: 100**

**Ex: 13 Dt:**

**Aim:**

Create a linked stack of college bus {bus\_no, from, to, start\_time, stops (2d array)}.

Search for buses running in a specific route.

**Program:**

#include<stdio.h>

#include<conio.h>

void main()

{ struct node

{ char b\_no[11],frm[20],to[20],ti[10],stp[5][20];

struct node \* next;

}\*start=NULL ,\*t,\*temp;

int i,c,ct,ch,s;

char from[20],to1[20],stops[5][20],route1[20],route2[20],time[10],no[11];

while(1)

{ system("CLS");

printf("1.INSERT\n2.SEARCH\n3.TRAVERSE\n4.EXIT"); scanf("%d",&ch);

switch(ch)

{ case 1: t=(struct node \*)malloc(sizeof(struct node));

printf("BUS\_NO: "); scanf("%s",&no);

printf("FROM: "); scanf("%s",&from);

printf("TO: "); scanf("%s",&to1);

printf("START\_TIME: "); scanf("%s",&time);

printf("STOPS \n");

for(i=0;i<5;i++)

{ scanf("%s",&stops[i]); }

for(i=0;i<5;i++)

{ strcpy(t->stp[i],stops[i]); }

strcpy(t->frm,from); strcpy(t->to,to1);

strcpy(t->b\_no,no); strcpy(t->ti,time);

if(start==NULL)

{ start=t; t->next=NULL; }

else

{ t->next=start; start=t; }

break;

case 2: if(start==NULL)

{ printf("NO DATA THERE!"); }

else

{ printf("GIVE ROUTE: "); scanf("%s%s",&route1,&route2);

temp=start; s=0;

while(temp!=NULL)

{ for(i=0;i<5;i++)

{ if(strcmp(temp->stp[i],route1)==0)

{for(i=0;i<5;i++)

{ if(strcmp(temp->stp[i],route2)==0)

{ printf("BUS\_NO: %s",temp->b\_no); s++; }

}//forloop

}//if

}//for

temp=temp->next;

}//while

if(s==0)

{ printf("NO BUS IS THERE FOR GIVEN ROUTE"); }

else

{ printf("\n%d BUS FOR GIVEN ROUTE !!",s); }

}//else

break;

case 3: if(start==NULL)

{ printf("NO DATA THERE!!"); }

else

{ temp=start; system("CLS");

while(temp!=NULL)

{ printf("BUS\_NO: %s",temp->b\_no);

printf("\nSTOPS");

for(i=0;i<5;i++)

{ printf("\n%s",temp->stp[i]); }

printf("\nFROM: %s",temp->frm);

printf("\nTO: %s",temp->to);

printf("\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");

temp=temp->next;

}

}break;

case 4: exit(0);

default: printf("INVALID ENTRY!!");

}//switch

getch();

}//while

}//main

**Sample Input-Output:**

1. **INSERT**
2. **SEARCH**
3. **TRAVERSE**
4. **EXIT 1**

**BUS\_NO: 101**

**FROM: XYZ**

**TO: ABC**

**START\_TIME: 6:42**

**STOPS**

1. 122 121 120 124
2. **INSERT**
3. **SEARCH**
4. **TRAVERSE**
5. **EXIT 1**

**BUS\_NO: 102**

**FROM: ESC**

**TO: CSE**

**START\_TIME: 3:33**

**STOPS**

ABC ESC CSE XYZ QWERTY

1. **INSERT**
2. **SEARCH**
3. **TRAVERSE**
4. **EXIT 2**

**GIVE ROUTE: ESC QWERTY**

**BUS\_NO: 102**

1. **BUS FOR GIVEN ROUTE !!**
2. **INSERT**
3. **SEARCH**
4. **TRAVERSE**
5. **EXIT 3**

**BUS\_NO: 102**

**FROM: ESC**

**TO: CSE**

**STOPS**

ABC ESC CSE XYZ QWERTY

**BUS\_NO: 101**

**FROM: XYZ**

**TO: ABC**

**STOPS**

123 122 121 120 124

1. **INSERT**
2. **SEARCH**
3. **TRAVERSE**
4. **EXIT 4**

**Ex: 14 Dt:**

**Aim:**

Create a linked queue of trains {train\_no, from, to, distance, Date\_wise\_ availability (2d

array)}. Perform insert, delete, display and find availability of train based on travel date.

**Program:**

#include<stdio.h>

#include<conio.h>

void main()

{ struct node

{ char train\_no[20];

char frm[20],to[20];

char date\_a[5][10];

char distance[10];

struct node \* next;

}\*start=NULL ,\*t,\*temp;

int i,c,ct,ch,s;

char from[20],to1[20],a[5][10],no[11],d[10],dd[10],fr[20],to2[20];

while(1)

{ system("CLS");

printf("1.INSERT\n2.DELETE\n3.TRAVERSE\n4.SEARCH\n5.EXIT");

scanf("%d",&ch);

system("CLS");

switch(ch)

{ case 1: t=(struct node \*)malloc(sizeof(struct node));

printf("TRAIN\_NO: ");

scanf("%s",&no);

printf("FROM: ");

scanf("%s",&from);

printf("TO: ");

scanf("%s",&to1);

printf("DISTANCE: ");

scanf("%s",&d);

printf("DATE\_WISE\_AVAILABITY\n");

for(i=0;i<5;i++)

{ scanf("%s",&a[i]); }

for(i=0;i<5;i++)

{ strcpy(t->date\_a[i],a[i]); }

strcpy(t->frm,from);

strcpy(t->to,to1);

strcpy(t->train\_no,no);

strcpy(t->distance,d);

if(start==NULL)

{ start=t; t->next=NULL; }

else

{ temp=start;

while(temp->next!=NULL)

{ temp=temp->next; } temp->next=t; t->next=NULL;

}break;

case 2: if(start==NULL)

{ printf("NO DATA THERE!!"); }

else if(start->next==NULL)

{ t=start; free(t); start=NULL; }

else

{ t=start; start=start->next; free(t); }

break;

case 3: if(start==NULL)

{ printf("NO DATA THERE!!"); }

else

{ temp=start;

while(temp!=NULL)

{ printf("TRAIN\_NO: %s",temp->train\_no);

printf("\nDATE\_WISE\_AVAILABITY");

for(i=0;i<5;i++)

{ printf("\n%s",temp->date\_a[i]); }

printf("\nFROM: %s",temp->frm);

printf("\nTO: %s",temp->to);

printf("\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");

temp=temp->next;

}

}break;

case 4: if(start==NULL)

{ printf("empty"); }

else

{ printf("INSERT\_DATE: ");

scanf("%s",&dd);

printf("FROM: ");

scanf("%s",&fr);

printf("TO: ");

scanf("%s",&to2);

temp=start; s=0;

while(temp!=NULL)

{ if((strcmp(temp->frm,fr)==0) && (strcmp(temp->to,to2)==0))

{ for(i=0;i<5;i++)

{ if(strcmp(temp->date\_a[i],dd)==0)

{ s++; }

}

} temp=temp->next;

}

if(s==0)

{ printf("DATA NOT FOUND"); }

else

{ printf("%d TRAINS AVAILABLE ON GIVEN DATE",s); }

}break;

case 5: exit(0);

default: printf("INVALID ENTRY!!");

}//switch

getch();

}//while

}//main

**Sample Input-Output:**

1. **INSERT**
2. **DELETE**
3. **TRAVERSE**
4. **SEARCH**
5. **EXIT 1**

**TRAIN\_NO: 101**

**FROM: ABC**

**TO: ESC**

**DISTANCE: 100**

**DATE\_WISE\_AVAILABITY**

**12**

**14**

**16**

**18**

**20**

1. **INSERT**
2. **DELETE**
3. **TRAVERSE**
4. **SEARCH**
5. **EXIT 4**

**INSERT\_DATE: 20**

**FROM: ABC**

**TO: ESC**

1. **TRAINS AVAILABLE ON GIVEN DATE**
2. **INSERT**
3. **DELETE**
4. **TRAVERSE**
5. **SEARCH**
6. **EXIT 5**

**Ex: 15 Dt:**

**Aim:**

Create a doubly linked list of students {rollno, name, event\_name} who wish to

participate in different sports events. Take a specific student name and find the students

who are participating in the same event by traversing both sides in the list.

**Program:**

#include<stdio.h>

#include<conio.h>

#include<string.h>

void main()

{ struct node

{

struct node \*next;

struct node \*back;

char roll\_no[11];

char name[20];

char event\_name[20];

}\*start=NULL,\*t,\*temp;

int ch,s,pos,c,ct;

char r[11],n[20],nm[20];

while(1)

{ system("CLS");

printf("1.INSERT\n2.search\n3.traverse\n4.EXIT");

scanf("%d",&ch); system("CLS");

switch(ch)

{

case 1: t=(struct node \*)malloc(sizeof(struct node));

printf("ROLL\_NO: "); scanf("%s",&r);

printf("NAME: "); scanf("%s",&n);

printf("EVENT\_NAME: "); scanf("%s",&nm);

strcpy(t->roll\_no,r);

strcpy(t->name,n);

strcpy(t->event\_name,nm);

if(start==NULL)

{ t->next=NULL; t->back==NULL; start=t; }

else

{ temp=start;

while(temp->next!=NULL)

{ temp=temp->next; }

temp->next=t; t->back=temp; t->next=NULL;

}break;

case 2: if(start==NULL)

{ printf("DATA EMPTY!!"); }

else

{ printf("STUDENT\_NAME: "); scanf("%s",&n);

s=0; t=start;

while(t!=NULL)

{ if(strcmp(t->name,n)==0)

{ strcpy(nm,t->event\_name);

s++; break;

}

t=t->next;

}

if(s==0)

{ printf("NOT FOUND!!!"); }

else

{ printf("\nSTUDENT\_WITH\_SAME\_EVENT");

t=start;

while(t!=NULL)

{ if(strcmp(t->event\_name,nm)==0)

{ printf("\n%s",t->name); }

t=t->next;

}

}

}break;

case 3: if(start==NULL)

{ printf("EMPTY DATA!!!"); }

else

{ temp=start;

while(temp!=NULL)

{ printf("\nNAME: %s",temp->name);

printf("\nROLL\_NO: %s",temp->roll\_no);

printf("\nEVENT\_NAME: %s",temp->event\_name);

printf("\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");

temp=temp->next;

}

}break;

case 4: exit(0);

}

getch();

}

}

**Sample Input-Output:**

1. **INSERT**
2. **SEARCH**
3. **TRAVERSE**
4. **EXIT 1**

**ROLL\_NO: 123**

**NAME: YUVRAJ**

**EVENT\_NAME: ABC**

1. **INSERT**
2. **SEARCH**
3. **TRAVERSE**
4. **EXIT 1**

**ROLL\_NO: 122**

**NAME: QWERTY**

**EVENT\_NAME: ABC**

1. **INSERT**
2. **SEARCH**
3. **TRAVERSE**
4. **EXIT 1**

**ROLL\_NO: 121**

**NAME: XYZ**

**EVENT\_NAME: ESC**

1. **INSERT**
2. **SEARCH**
3. **TRAVERSE**
4. **EXIT 2**

**STUDENT\_NAME: YUVRAJ**

**STUDENT\_WITH\_SAME\_EVENT**

**YUVRAJ**

**QWERTY**

1. **INSERT**
2. **SEARCH**
3. **TRAVERSE**
4. **EXIT 3**

**NAME: YUVRAJ**

**ROLL\_NO: 123**

**EVENT\_NAME: ABC**

**NAME: QWERTY**

**ROLL\_NO: 122**

**EVENT\_NAME: ABC**

**NAME: XYZ**

**ROLL\_NO: 121**

**EVENT\_NAME: ESC**

1. **INSERT**
2. **SEARCH**
3. **TRAVERSE**
4. **EXIT 4**

**Ex: 16 Dt:**

**Aim:**

Take the list of vegetables {name, price, vendor}. Search for a specific vegetable and list

the vendors ordered by price using selection sort.

**Program:**

#include<stdio.h>

#include<conio.h>

void main()

{ struct node

{ char v\_name[20];

int price[5],qt;

char vg\_n[5][10];

struct node \* next;

}\*start=NULL ,\*t,\*temp;

int i,c,ct,ch,s,pr[5];

char v\_nam[20],nam[5][10],vg[10];

while(1)

{ system("CLS");

printf("1.INSERT\n2.TRAVERSE\n3.SEARCH\n4.EXIT");

scanf("%d",&ch);

system("CLS");

switch(ch)

{ case 1: t=(struct node \*)malloc(sizeof(struct node));

printf("VENDOR\_NAME: ");

scanf("%s",&v\_nam);

for(i=1;i<=5;i++)

{ printf("VEGETABLE\_NAMES: ");

scanf("%s",&nam[i]);

printf("PRICE: ");

scanf("%d",&pr[i]);

}

strcpy(t->v\_name,v\_nam);

for(i=1;i<=5;i++)

{ strcpy(t->vg\_n[i],nam[i]);

t->price[i]=pr[i];

}

if(start==NULL)

{ start=t; t->next=NULL; }

else

{ temp=start;

while(temp->next!=NULL)

{ temp=temp->next; } temp->next=t; t->next=NULL;

}break;

case 2: if(start==NULL)

{ printf("NO DATA THERE!!"); }

else

{ temp=start;

while(temp!=NULL)

{ printf("VENDOR\_NAME: %s",temp->v\_name);

for(i=1;i<=5;i++)

{ printf("\nVEGETABLE\_NAME: %s",temp->vg\_n[i]);

printf("\nPRICE: %d",temp->price[i]);

}

printf("\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");

temp=temp->next;

}

}break;

case 3: if(start==NULL)

{ printf("empty"); }

else

{ printf("VEGETABLE\_NAME: ");

scanf("%s",&vg);

temp=start; s=0;

while(temp!=NULL)

{ for(i=1;i<=5;i++)

{ if(strcmp(temp->vg\_n[i],vg)==0)

{ s++;

printf("\n%d.VENDORS: %s",s,temp->v\_name);

printf("\nPRICE: %d",temp->price[i]);

}

}

temp=temp->next;

}

if(s==0)

{ printf("DATA NOT FOUND"); }

else

{ printf("\nSELECTION OF VENDORS:");

scanf("%s",&v\_nam);

c=1; temp=start;

while(c)

{ if(strcmp(temp->v\_name,v\_nam)==0)

{ for(i=1;i<=5;i++)

{ if(strcmp(temp->vg\_n[i],vg)==0)

{ printf("QUANTITY:");

scanf("%d",&temp->qt);

temp->qt=temp->price[i]\*temp->qt;

printf("TOTAL\_PRICE=%d",temp->qt);

printf("\nORDER\_DONE!!");

}

}c=0;

}

else

{ temp=temp->next; }

}

}

}break;

case 4: exit(0);

default: printf("INVALID ENTRY!!");

}//switch

getch();

}//while

}//main

**Sample Input-Output:**

1. **INSERT**
2. **TRAVERSE**
3. **SEARCH**
4. **EXIT 1**

**VENDOR\_NAME: 1234**

**VEGETABLE\_NAME: A**

**PRICE: 10**

**VEGETABLE\_NAME: B**

**PRICE: 11**

**VEGETABLE\_NAME: C**

**PRICE: 12**

**VEGETABLE\_NAME: D**

**PRICE: 13**

**VEGETABLE\_NAME: E**

**PRICE: 14**

1. **INSERT**
2. **TRAVERSE**
3. **SEARCH**
4. **EXIT 1**

**VENDOR\_NAME: 123**

**VEGETABLE\_NAME: A**

**PRICE: 9**

**VEGETABLE\_NAME: B**

**PRICE: 8**

**VEGETABLE\_NAME: C**

**PRICE: 12**

**VEGETABLE\_NAME: D**

**PRICE: 13**

**VEGETABLE\_NAME: E**

**PRICE: 14**

1. **INSERT**
2. **TRAVERSE**
3. **SEARCH**
4. **EXIT 3**

**VEGETABLE\_NAME: A**

1. **VENDORS: 1234**

**PRICE: 10**

1. **VENDORS: 123**

**PRICE: 9**

**SELECTION OF VENDOR: 123**

**QUANTITY: 10**

**TOTAL\_PRICE=90**

**ORDER\_DONE!!**

1. **INSERT**
2. **TRAVERSE**
3. **SEARCH**
4. **EXIT 2**

**VENDOR\_NAME: 1234**

**VEGETABLE\_NAME: A**

**PRICE: 10**

**VEGETABLE\_NAME: B**

**PRICE: 11**

**VEGETABLE\_NAME: C**

**PRICE: 12**

**VEGETABLE\_NAME: D**

**PRICE: 13**

**VEGETABLE\_NAME: E**

**PRICE: 14**

**VENDOR\_NAME: 123**

**VEGETABLE\_NAME: A**

**PRICE: 9**

**VEGETABLE\_NAME: B**

**PRICE: 8**

**VEGETABLE\_NAME: C**

**PRICE: 12**

**VEGETABLE\_NAME: D**

**PRICE: 13**

**VEGETABLE\_NAME: E**

**PRICE: 14**

1. **INSERT**
2. **TRAVERSE**
3. **SEARCH**
4. **EXIT 4**

**Ex: 17 Dt:**

**Aim:**

Write a program to take the details of patients {name, token\_no}. Create an insert\_sort

function that takes elements one by one and insert them at proper position. Display each

time after inserting the elements.

**Program:**

#include<stdio.h>

#include<conio.h>

#include<string.h>

void main()

{ struct node

{ struct node \*next;

struct node \*back;

char name[20]; int token\_no;

}\*start=NULL,\*t,\*temp,\*temps;

int ch,s,no; char n[20];

while(1)

{ printf("1.INSERT\n2.EXIT");

scanf("%d",&ch); system("CLS");

switch(ch)

{case 1: t=(struct node \*)malloc(sizeof(struct node));

printf("NAME: "); scanf("%s",&n);

printf("TOKEN\_NO: "); scanf("%d",&no);

strcpy(t->name,n); t->token\_no=no;

if(start==NULL)

{ t->next=NULL; t->back==NULL; start=t; }

else if(start->token\_no>t->token\_no)

{ t->back=NULL;

t->next=start;

start->back=t;

start=t;

}

else

{ temp=start; s=0;

while(temp!=NULL)

{ if(temp->token\_no>t->token\_no)

{ t->back=temp->back;

temp->back=t;

temps->next=t;

t->next=temp;

s++;

break;

}temps=temp;

temp=temp->next;

}

if(s==0)

{ temps->next=t; t->back=temps; t->next=NULL; }

}

if(start==NULL)

{ printf("EMPTY DATA!!!"); }

else

{ temp=start;

while(temp!=NULL)

{ printf("\nNAME: %s",temp->name);

printf("\nTOKEN\_NO: %d",temp->token\_no);

printf("\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");

temp=temp->next;

}

}break;

case 2: exit(0);

default: printf("INVALID ENTRY!!");

}

getch();

}

}

**Sample Input-Output:**

1. **INSERT**
2. **EXIT 1**

**NAME: 123**

**TOKEN\_NO: 123**

**NAME: 123**

**TOKEN\_NO: 123**

1. **INSERT**
2. **EXIT 1**

**NAME: 122**

**TOKEN\_NO: 122**

**NAME: 122**

**TOKEN\_NO: 122**

**NAME: 123**

**TOKEN\_NO: 123**

**Ex: 18 Dt:**

**Aim:**

Write a program to sort the details of vehicles {vehicle\_no, type, toll\_amount (1D array),toll\_name(2D array)} based on toll\_amount using Bubble sort.

**Program:**

#include<stdio.h>

#include<conio.h>

void main()

{ struct node

{ struct node \*next;

char v\_no[20],type[20];

struct toll

{ char name[20];

int amount;

}toll[5],tl;

}\*start=NULL,\*t,\*temp;

int ch,i,j;

while(1)

{ system("CLS");

printf("1.INSERT\n2.DISPLAY\n3.EXIT"); scanf("%d",&ch);

switch(ch)

{ case 1: t=(struct node \*)malloc(sizeof(struct node));

printf("VEHICLE\_NO: "); scanf("%s",&t->v\_no);

printf("TYPE: "); scanf("%s",&t->type);

for(i=0;i<5;i++)

{ printf("TOLL\_NAME & TOLL\_AMOUNT\n");

scanf("%s%d",&t->toll[i].name,&t->toll[i].amount);

if(i>1)

{ for(j=0;j<i;j++)

{ if(t->toll[i].amount<t->toll[j].amount)

{ t->tl=t->toll[i]; t->toll[i]=t->toll[j]; t->toll[j]=t->tl; }

} }}

if(start==NULL)

{ start=t; t->next=NULL; }

else

{ temp=start;

while(temp->next!=NULL)

{ temp=temp->next; } temp->next=t; t->next=NULL;

}break;

case 2: if(start==NULL)

{ printf("NO DATA THERE!!"); }

else

{ temp=start; system("CLS");

while(temp!=NULL)

{ printf("VEHICLE\_NO: %s",temp->v\_no);

for(i=0;i<5;i++)

{ printf("\n%s\t%d",t->toll[i].name,t->toll[i].amount);}

printf("\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");

temp=temp->next;

}

}break;

case 3: exit(0);

default: printf("invalid");

}getch();

}

}

**Sample Input-Output:**

1. **INSERT**
2. **DISPLAY**
3. **EXIT 1**

**VEHICLE\_NO: 101**

**TYPE: QWERTY**

**TOLL\_NAME & TOLL\_AMOUNT**

**E 50**

**TOLL\_NAME & TOLL\_AMOUNT**

**C 25**

**TOLL\_NAME & TOLL\_AMOUNT**

**D 35**

**TOLL\_NAME & TOLL\_AMOUNT**

**B 20**

**TOLL\_NAME & TOLL\_AMOUNT**

**A** **10**

1. **INSERT**
2. **DISPLAY**
3. **EXIT 2**

**VEHICLE\_NO: 101**

**TOLL\_NAME & TOLL\_AMOUNT**

**A 10**

**B 20**

**C 25**

**D 35**

**E 50**

1. **INSERT**
2. **DISPLAY**
3. **EXIT 3**

**Ex: 19 Dt:**

**Aim:**

Write a program in C to create a dictionary structure {word, meaning, opposite}. Sort

them and perform binary search to find a specific word. Allow adding new words. Also

display the most frequently searched word.

**Program:**

#include<stdio.h>

#include<conio.h>

void main()

{ struct node

{ struct node \*next;

struct node \*back;

char word[20],mean[50],opp[20];

}\*start=NULL,\*t,\*temp,\*temps;

int ch,s,i=0;

char w[20],f\_w[5][20];

while(1)

{system("CLS"); printf("1.INSERT\n2.DISPLAY\n3.SEARCH\n4.FRQUENTLY\_SEARCH\_WORD\n5.EXIT");

scanf("%d",&ch);

switch(ch)

{ case 1: t=(struct node \*)malloc(sizeof(struct node));

printf("ENTER\_WORD: "); scanf("%s",&t->word);

printf("MEANING: "); scanf("%s",&t->mean);

printf("OPPOSITE\_WORD: "); scanf("%s",&t->opp);

if(start==NULL)

{ t->next=NULL; t->back=NULL; start=t; }

else if(strcmp(start->word,t->word)>0)

{ t->next=start; t->back=NULL; start->back=t; start=t; }

else

{ temp=start; s=0;

while(temp!=NULL)

{ if(strcmp(temp->word,t->word)>0)

{ t->next=temp; t->back=temp->back; temp->back=t;

temps->next=t; s++; break; }

temps=temp; temp=temp->next;

}

if(s==0)

{ temps->next=t; t->back=temps; t->next=NULL; }

}break;

case 2: if(start==NULL)

{ printf("NO DATA THERE!!"); }

else

{ temp=start; system("CLS");

while(temp!=NULL)

{ printf("WORD: %s",temp->word);

printf("\nMEANING: %s",temp->mean);

printf("\nOPPOSITE: %s",temp->opp);

printf("\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");

temp=temp->next;

}

}break;

case 3: if(start==NULL)

{ printf("NO DATA THERE!"); }

else

{ printf("GIVE WORD: "); scanf("%s",&w);

temp=start; s=0;

while(temp!=NULL)

{ if(strcmp(temp->word,w)==0)

{ printf("WORD: %s",temp->word);

printf("\nMEANING: %s",temp->mean);

printf("\nOPPOSITE: %s",temp->opp);

printf("\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");

strcpy(f\_w[i],w); i++; s++;

break;

}temp=temp->next;

}

if(s==0)

{ printf("NOT\_FOUND"); }

}break;

case 4: printf("FREQUENTLY\_SEARCH\_WORDS");

for(s=0;s<i;s++)

{ printf("\n%s",f\_w[s]); }

break;;

case 5: exit(0);

default: printf("invalid");

}getch();

}

}

**Sample Input-Output:**

1. **INSERT**
2. **DISPLAY**
3. **SEARCH**
4. **FREQUENTLY\_SEARCH\_WORD**
5. **EXIT 1**

**ENTER WORD: ABC**

**MEANING: ABC**

**OPPOSITE\_WORD: ABC**

1. **INSERT**
2. **DISPLAY**
3. **SEARCH**
4. **FREQUENTLY\_SEARCH\_WORD**
5. **EXIT 1**

**ENTER WORD: XYZ**

**MEANING: ABC**

**OPPOSITE\_WORD: ABC**

1. **INSERT**
2. **DISPLAY**
3. **SEARCH**
4. **FREQUENTLY\_SEARCH\_WORD**
5. **EXIT 1**

**ENTER WORD: ESC**

**MEANING: ABC**

**OPPOSITE\_WORD: ABC**

1. **INSERT**
2. **DISPLAY**
3. **SEARCH**
4. **FREQUENTLY\_SEARCH\_WORD**
5. **EXIT 1**

**ENTER WORD: QWERTY**

**MEANING: ABC**

**OPPOSITE\_WORD: ABC**

1. **INSERT**
2. **DISPLAY**
3. **SEARCH**
4. **FREQUENTLY\_SEARCH\_WORD**
5. **EXIT 1**

**ENTER WORD: CSE**

**MEANING: ABC**

**OPPOSITE\_WORD: ABC**

1. **INSERT**
2. **DISPLAY**
3. **SEARCH**
4. **FREQUENTLY\_SEARCH\_WORD**
5. **EXIT 2**

**WORD: ABC**

**MEANING: ABC**

**OPPOSITE: ABC**

**WORD: CSE**

**MEANING: ABC**

**OPPOSITE: ABC**

**WORD: ESC**

**MEANING: ABC**

**OPPOSITE: ABC**

**WORD: QWERTY**

**MEANING: ABC**

**OPPOSITE: ABC**

**WORD: XYZ**

**MEANING: ABC**

**OPPOSITE: ABC**

1. **INSERT**
2. **DISPLAY**
3. **SEARCH**
4. **FREQUENTLY\_SEARCH\_WORD**
5. **EXIT 3**

**GIVE WORD: CSE**

**MEANING: ABC**

**OPPOSITE\_WORD: ABC**

1. **INSERT**
2. **DISPLAY**
3. **SEARCH**
4. **FREQUENTLY\_SEARCH\_WORD**
5. **EXIT 4**

**FREQUENTLY\_SEARCH\_WORDS**

**CSE**

1. **INSERT**
2. **DISPLAY**
3. **SEARCH**
4. **FREQUENTLY\_SEARCH\_WORD**
5. **EXIT 5**