C Programs

Coding 2D array with pointer

1.

```
#include <stdio.h>
 int
main ()
{
int ar[2][2];
ar[0][0] = 33;
ar[0][1] = 22;
ar[1][0] = 11;
ar[1][1] = 10;
printf ("%u\n", **(ar));
printf ("%u\n", **(ar+0));
printf ("%u\n", *(*(ar+0)+0));
printf ("%u\n", *(*(ar+0)+1));
printf ("%u\n", **(ar+1));
printf("%u\n",*(*(ar+1)+1));
}
```

2.

```
#include <stdio.h>
main ()
{
int ar[2][2];
ar[0][0] = 33;
ar[0][1] = 22;
ar[1][0] = 11;
ar[1][1] = 10;
```

```
printf ("%u\n", (ar));
printf ("%u\n", (ar+0));
printf ("%u\n", ((ar+0)+0));
printf ("%u\n", (ar+0)+1);
printf ("%u\n", (ar)+1);
printf("%u\n",((ar+1)+1));
}
```

3. Scope of variable;

```
#include <stdio.h>
int a=10;// global scope
int main ()
{
   int b=11;//local scope
   {
      int c=12;// local scope
      printf("%d ",c);
      printf("%d ",a);
   }

}
```

4.Call by reference:

```
#include <stdio.h>
#include<string.h>
int main(){
   int a=10;
   printf("before %d\n",a);
```

```
    change(&a);
    printf("after %d\n",a);
   return 0;
}
void change(int *p)
{
    *p=*p+100; //

}
```

5.Malloc Program

```
#include <stdio.h>
#include <stdlib.h>
main ()
{int n, i, *ptr, sum = 0;
printf ("Enter number of elements: ");
scanf ("%d", &n);
ptr = (int *) malloc (n * sizeof (int));
    // if memory cannot be allocated
    if (ptr == NULL)
    {   printf ("Error! memory not allocated.");
        exit (0);
    }
printf ("Enter elements: ");
for (i = 0; i < n; ++i)
    {scanf ("%d", ptr + i);
        sum += (ptr[i]);
    }
printf ("Sum = %d\n", sum);
printf ("address of Sum= %u\n", &sum);
  // deallocating the memory
free (ptr);
```

```
}
```

Malloc with 2d arrays:

```
#include <stdio.h>
#include<stdlib.h>
int main()
{   int ar[2][3];
    int **p;
    p = ((int *) malloc(sizeof(int)*2));
     *(p+0)=((int *) malloc(sizeof(int)*3));
      *(p+1)=((int *) malloc(sizeof(int)*3));
or
p = ((int *)malloc(sizeof(int)*2));
    for(int i=0;i<2;i++)
    {
        p[i]=(int *)malloc(sizeof(int)*3);
    }
      *(*(p+0)+0)=100;
      *(*(p+0)+1)=200;
      *(*(p+0)+2)=300;
      *(*(p+1)+0)=400;
      *(*(p+1)+1)=500;
      *(*(p+1)+2)=600;
      //Printing the address
      printf("%d ",&p[0][0]);
      printf("%d ",&p[0][1]);
      printf("%d ",&p[0][2]);
      printf("\n%d ",&p[1][0]);
      printf("%d ",&p[1][1]);
      printf("%d ",&p[1][2]);
      //Printing the values.
```

```
    printf("\n%d ",p[0][0]);
    printf("%d ",p[0][1]);
    printf("%d ",p[0][2]);
    printf("\n%d ",p[1][0]);
    printf("%d ",p[1][1]);
    printf("%d ",p[1][2]);
}
```

Malloc with 1d array

```
#include<stdio.h>
int main ()
{
 int a[5]; // static allocation
 a[0]=100;
 a[1]=200;

 int *p;
 p = (int *) malloc(sizeof(int)*5);// Dynamic allocation

 *(p+0)=100; //p[0]=100;
 *(p+1)=200;
 printf("%d\n",*(p+0));
 printf("%d\n",*(p+1));
}
```

Malloc with single variable:

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

```
int main()
{
    int a=10;
    int *p;
    p=(int *) malloc(sizeof(int));
    *p=10;
    printf("%d\n",*p);

    return 0;
}
```

Structure with pointers program

```
#include<stdio.h>
#include<stdlib.h>
struct customer
{
  char name[20];
  char id[20];
};
int
main ()
{
  struct customer *pc1;
  int i, n;
  printf ("enter no of customers\n");
  scanf ("%d", &n);
  pc1 = (struct customer *) malloc (n * sizeof ( struct customer));
  for (i = 0; i < n; i++)
    {
      printf ("enter the values of name and id\n");
```

```
      scanf ("%s %s", (pc1 + i)->name, (pc1 + i)->id);
      //or scanf ("%s %s", (pc1[i)].name, (pc1 + i)->id);
    }
  for (i = 0; i < n; i++)
    {
      printf ("%s %s", (pc1 + i)->name, (pc1 + i)->id);
    }
}
```

Swaping of two numbers using call by reference;

```
#include<stdio.h>
int main()
{
    int a=10,b=20;
    printf("Before swaping a,b= %d,%d\n",a,b);
    swap(&a,&b);
     printf("After swaping a,b= %d,%d\n",a,b);
}
void swap(int *a1,int *b1)
{
    int temp;
    temp=*a1;
    *a1=*b1;
    *b1=temp;
    printf("After swaping a,b= %d,%d\n",*a1,*b1);
}
```

Updating record in structure:

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

```
struct emp // new user defined datatype
 {
 char id[10];
char name[20];
 };
int main()
{
 struct emp employees[3]; // struct emp vs emp1
 char sid[10];
 char sname[20];
 int flag=0,j;

 printf("Enter 3 employee records id and name \n");

 for (int i = 0; i < 3; i++) {
 scanf("%s",employees[i].id);
 scanf("%s",employees[i].name);
 }

 for (int i = 0; i < 3; i++) {
 printf("%s\t",employees[i].id);
 printf("%s\n",employees[i].name);
 }

 printf("Enter employe id to update\n");
 scanf("%s",sid);

 for (int i = 0; i < 3; i++) {
 if(strcmp(sid,employees[i].id) == 0)
 {
    flag=1;
    j=i;
```

```
    break;
 }

}
if(flag==1)
{
    printf("Enter the new values");
    scanf("%s",sid);
    scanf("%s",sname);
    strcpy(employees[j].id,sid);
    strcpy(employees[j].name,sname);
    printf("Records after updating\n");
    for (int i = 0; i < 3; i++) {
 printf("%s\t",employees[i].id);
 printf("%s\n",employees[i].name);
 }
}
else
printf("Invalid data ");

}
```

Deleting record in structure

```
#include <stdio.h>
#include <string.h>
struct emp // new user defined datatype
 {
 char id[10];
 char name[20];
```

```
};

int main()
{
 struct emp employees[3]; // struct emp vs emp1
 char sid[10];
 int flag=0,j;
  char nullStr[20] = {"\0"};

 printf("Enter 3 employee records id and name \n");

 for (int i = 0; i < 3; i++) {
 scanf("%s",employees[i].id);
 scanf("%s",employees[i].name);
 }

 for (int i = 0; i < 3; i++) {
 printf("%s\t",employees[i].id);
 printf("%s\n",employees[i].name);
 }

 printf("Enter employe id to delete \n");
 scanf("%s",sid);

 for (int i = 0; i < 3; i++) {
 if(strcmp(sid,employees[i].id) == 0)
 {
 flag=1;
    strcpy(employees[i].name,nullStr);
    strcpy(employees[i].id,nullStr);
 break;
 }
```

```
}
if(flag==1)
{
    printf("data deleted succesfully\n");
     printf("Records after deleting\n");
 for (int i = 0; i < 3; i++) {
 printf("%s\t",employees[i].id);
 printf("%s\n",employees[i].name);
 }
}
else
printf("Invalid data ");

}
```

Program to perform multiple operations on structure:

```
#include <stdio.h>
#include <string.h>
struct emp // new user defined datatype
 {
 char id[10];
char name[20];
 };
 int z;
 char sid[10];
 char sname[20];
 int flag=0,j;
 int op;
```

```
 char nullStr[20] = {"\0"};
 struct emp employees[100];
 int currentindex=0;
int main()
{
  char nullStr[20] = {"\0"};
  while(1)
  {
  printf("Select  One Option");

printf("\n1.Addrecord\n2.Update\n3.Search\n4.Show\n5.Delete\n6.Exit
");
  scanf("\n%d",&op);
  switch(op)
  {
      case 1:
      Add();
      break;
      case 2:
      update();
      break;
      case 3:
      search();
      break;
      case 4:
      show();
      break;
      case 5:
      Delete();
      break;
      case 6:
      exit(0);
```

```
      default : printf("wrong choice");
  }

  }
}
void Add( )
{
     //printf("Enter Number of records to add \n");
     //scanf("%d",&z);
     printf("Enter ID and Name\n");
     //for (int i = 0; i < z; i++)
     //{
         scanf("%s",employees[currentindex].id);
         scanf("%s",employees[currentindex].name);
         currentindex++;
     //}
}
void show()
{
    for (int i = 0; i <currentindex; i++)
    {
        printf("%s\n",employees[i].id);
        printf("%s\n",employees[i].name);
    }
}
void update()
{
    printf("Enter employe id to update\n");
    scanf("%s",sid);

 for (int i = 0; i < 3; i++)
 {
```

```
    if(strcmp(sid,employees[i].id) == 0)
    {
        flag=1;
        j=i;
        break;
    }
 }
if(flag==1)
{
    printf("Enter the new values\n");
    scanf("%s",sid);
    scanf("%s",sname);
    strcpy(employees[j].id,sid);
    strcpy(employees[j].name,sname);
    printf("Records Updated Succesfully\n");
    for (int i = 0; i < 3; i++) {
 printf("%s\n",employees[i].id);
 printf("%s\n",employees[i].name);
 }
}
else
printf("Invalid data\n");

}
void Delete()
{
 printf("Enter employe id to delete \n");
 scanf("%s",sid);

 for (int i = 0; i < 3; i++) {
 if(strcmp(sid,employees[i].id) == 0)
 {
```

```
    flag=1;
    strcpy(employees[i].name,nullStr);
    strcpy(employees[i].id,nullStr);
    break;
 }
 }
if(flag==1)
{
    printf("data deleted succesfully\n");
    printf("Records after deleting\n");
 for (int i = 0; i < 3; i++)
 {
    printf("%s\t",employees[i].id);
    printf("%s\n",employees[i].name);
 }
}
else
printf("Invalid data\n");
}
void search(){
    printf("Enter employe id to search \n");
 scanf("%s",sid);

 for (int i = 0; i < 3; i++) {
 if(strcmp(sid,employees[i].id) == 0)
 {
    flag=1;
    j=i;
    break;
}
}
if(flag==1)
```

```
{
    printf("Employee Details\n");
    printf("%s\t",employees[j].id);
    printf("%s\n",employees[j].name);
}
else
printf("Invalid data\n");
}
```

3D array with pointers

```
#include <stdio.h>
#include<stdlib.h>
int main()
{   int ar[2][3][3][4];
    int*** p = (int *)malloc(2 * sizeof(int**));


    for (int i = 0; i < 2; i++)
    {
        p[i] = (int *)malloc(3 * sizeof(int*));

        for (int j = 0; j < 3; j++)
        {
            p[i][j] = (int*)malloc(3 * sizeof(int));

        }
    }

        *(*(*(p+0)+0)+0)=100;
      *(*(*(p+0)+0)+1)=200;
```

```
    *(*(*(p+0)+0)+2)=300;
    *(*(*(p+0)+1)+0)=400;
    *(*(*(p+0)+1)+1)=500;
    *(*(*(p+0)+1)+2)=600;
    *(*(*(p+1)+1)+0)=700;
    *(*(*(p+1)+1)+1)=800;
    *(*(*(p+1)+1)+2)=900;
    printf("\n%d ",p[0][0][0]);
    printf("%d ",p[0][0][1]);
    printf("%d ",p[0][0][2]);
    printf("\n%d ",p[0][1][0]);
    printf("%d ",p[0][1][1]);
    printf("%d ",p[0][1][2]);
    printf("\n%d ",p[0][1][0]);
    printf("%d ",p[1][1][0]);
    printf("%d ",p[1][1][2]);

}
```

SAMPLE STACK PROGRAM

```
// stack example with pointers
#include <stdio.h>
int stack[5],top=-1,n,size=5;
int main()
{
   while(1)
   {
      printf("\n1.push\n2.pop\n3.print\n4.exit\n");
       printf("Select your choice\n");
```

```
        scanf("%d",&n);
        switch(n)
        {
            case 1:
            push();
            break;
            case 2:
            pop();
            break;
            case 3:
            print();
            break;
            case 4:
            exit(0);

        }
    }

}
void push()
{
    int element;
    if(top == size-1)
    {
        printf("Stack is full");
    }
    else
    {
       printf("Entere element to push\n");
       scanf("%d ",&element);
       top++;
       stack[top]=element;
```

```
        printf("Inserted element\n");
        printf("%d",element);

    }

}
void pop()
{
    if (top==-1)
    {
        printf ("Stack is empty can't pop");

    }
    else
    {
        printf("poped element is %d\n", stack[top]);
        top--;
        return;
    }
}
void print()
{
    if(top==-1)
    {
        printf ("Stack is empty can't pop");


    }
    else
    {
        for(int i=top; i>=0;i--)
        {
```

```
        printf("%d " ,stack[top]);
      }
   }
}
```

LINKED LIST PROGRAM without pointers

```
#include <stdio.h>
#include<stdlib.h>
struct node
   {
      int data;
      struct node  *link; // self referential structure
   }l1,l2,l3,l4;

int main()
{
   l1.data=10;
   l2.data=13;
   l3.data=12;
   l4.data=11;
   l1.link=&l2;
   l2.link=&l3;
   l3.link=&l4;
   l4.link=NULL;
   printf("%d, %u ",l1.data,l1.link);
   printf("%d,%u ",l2.data,l2.link);
   printf("%d,%u ",l3.data,l3.link);
   printf("%u,%u ",l4.data,l4.link);
}
```

LINKED LIST WITH POINTERS----------

```
// stack example with pointers
#include <stdio.h>
#include<stdlib.h>
struct node
   {
      int data;
      struct node *link; // self refrential structured
   };
   struct node *first=NULL;
   int data1;
   int item;
   struct node *temp,*tmp;
int main()
{
   int n;
   while(1)
   {
      printf("Select your choice\n");
      printf("1.Insert beg\n2.Insert End\n3.Insert at Position\n4.Del
first\n5.Delete last\n6.Delete at
position\n7.print\n8.printpos\n9.exit\n");
      scanf("%d",&n);

      switch(n)
      {
         case 1:
         insertbeg();
         break;
         case 2:
         insertend();
         break;
```

```
            case 3:
            //insertend();
            break;
            //insertpos();
            break;
            case 4:
            //delbeg();
            break;
            case 5:
           // dellas();
            break;
            case 6:
            //delpos();
            break;
            case 7:
            print();
            break;
            case 8:
            //printpos();
            break;
            case 9:
            exit(0);
            break;

        }
    return 0;

    }

}
int insertbeg()
{
```

```
    if(first==NULL)
    {
        first=(struct node*)malloc(sizeof(struct node*));
        printf("Enter data for the node\n");
        scanf("%d ",&data1);
        first->data=data1;
        first->link=NULL;
    }
    else
    {
        temp=(struct node*)malloc(sizeof(struct node*));
        printf("Enter data for the node\n");
        scanf("%d ",&data1);
        temp->data=data1;
        temp->link=first;
        first=temp;
    }
    return 0;

}
void print()
{
    if(first == NULL)
    {
        printf("List is empty");
    }
    else
    {
        temp =first;
        while(temp !=NULL)
        {
            printf("[%d %u]\t",temp->data,temp->link);
```

```
            temp=temp->link;
        }
    }
}
void insertend()
{    struct node *temp,*l1;
    //struct node *l1,*tmp;
    l1=(struct node *)malloc(sizeof(struct node));
    temp=first;
    while(temp->link!=NULL)
    {
        temp=temp->link;
    }

    temp->link=l1;
    l1->link=NULL;
    printf("enter the data:\n");
    scanf("%d",&item);
    l1->data=item;

}

delend()
{
        Struct node *temp1;
        temp1=first;
        while(temp1->link!=NULL)
        {
                temp1=temp1->link;
        }
```

LINKED LIST WITH ALL THE OPERATIONS

```
#include <stdio.h>

struct node
{
 int data;
 struct node *link;
};
typedef struct node ll;

ll *first=NULL,*temp ;int data=0,choice=0,sizeoflist;
int main ()
{


while(choice != 9)

 {

 menu();

 scanf("%d",&choice);

 switch(choice)

 {

 case 1:
insertend();
 break;
```

```
case 2:
insertbeg();
break;
case 3:

insertatpos();

break;
case 4:
deleteatbeg();
break;
case 5: deletepos();

break;
case 6:
delend();
break;
case 7:print(); break;
case 8:
search();
break;
case 9: exit(0);
default : printf("Wrong choice");




}


}
```

```
return 0;

}

 void menu()

 {

 printf("\nEnter your choice\n");
 printf("1.Insert End \n2.Insert Beg \n3.Insert after which position \n");
 printf("4.Del First \n5.Delete Node no \n6.delend\n");
 printf("7.Print \n8.Search \n9.Exit \n");
 return;

 }



 void insertbeg()
 {
 if(first == NULL) // if the list is empty
 {
```

```
 first = (ll *)malloc(sizeof(ll));

 ++sizeoflist;
 printf("Enter data for the node \n");
 scanf("%d",&data);
 first->data = data;
 first->link = NULL;


 }

 else
 {
 temp = (ll *)malloc(sizeof(ll));
 printf("Enter data for the node \n");
 scanf("%d",&data);
 temp->data = data;
 temp->link = first;
 first = temp;
 }

 }

void insertatpos()
{
 int position=0; int tdata; ll * temp1;

 printf("Enter after which node you want to insert node ");
 scanf("%d",&position);

 if(first == NULL)
 {
```

```
 printf("Linked List Does not exit ");
 return;
 }
 if(position == 0)
 {

 insertbeg();
 return;
 }

 if(position > sizeoflist )
 {
 printf("Out of bounds ");
 return ;

 }//
 // 3
 temp = first;
 for (int i = 1; i < position; i++) {
 temp = temp->link;
 }
 printf("Enter data");
 scanf("%d",&tdata);
 temp1 = (ll *) malloc(sizeof(ll));
 temp1->link = temp->link;
 temp->link = temp1;
 temp1->data = tdata;
 ++sizeoflist;
}
void deleteatbeg()
{
 ll * temp1;
```

```
 if(first == NULL){
 printf("No List ");
 return ;
 }
 printf("Deleted Node Data is %d",first->data);
 temp1 = first;
 first= first->link;
 free(temp1);
}
void deletepos()
{
 ll * temp,*temp1; int pos;

 temp = first;
 printf("Enter Position to delete ");
 scanf("%d",&pos);
 if(pos == 0)
 {
 deleteatbeg();
 return;
 }

 for (int i = 1; i < pos; i++) {
 temp1 = temp;
 temp = temp -> link;
 }
 temp1->link = temp->link;
 printf("Deleted Data is %d \n",temp->data);
 free(temp);

}
void print(){
```

```
 if(first == NULL)
 {
 printf("List is empty ");
 }
 else
 {
 temp = first;
 while(temp != NULL)
 {
 printf(" [%d %u]-> \t",temp->data,temp->link);
 temp = temp -> link;
 }

 }

}

void search(){
 int searchelement=0,foundindex=0,searchposition,flag=0;
 ll * searchpointer,*stemp;
 if(first == NULL)
 {
 printf("List Empty");
 return;
 }

 stemp = first;
 printf("Enter element to search ");
 scanf("%d",&searchelement);
 for (int i = 1;stemp!=NULL ; i++) {
 if((stemp->data) == searchelement)
 {
```

```
 flag = 1 ;
searchpointer=stemp;
 searchposition = i;
 break;
 }
 stemp=stemp->link;
 }

 if(flag == 1)
 {
 printf("Found %d at %d \n",searchpointer->data,searchposition);
 }
 else
 printf("%d not found",searchelement);
}
void insertend()
{    struct node *temp,*l1;
int item;
    //struct node *l1,*tmp;
    l1=(struct node *)malloc(sizeof(struct node));
    temp=first;
    while(temp->link!=NULL)
    {
        temp=temp->link;
    }

    temp->link=l1;
    l1->link=NULL;
    printf("enter the data:\n");
    scanf("%d",&item);
    l1->data=item;
```

```
}
void delend()
{
        struct node *temp1,*temp2;
        temp2=first;
        while(temp2->link!=NULL)
        {
           temp1=temp2;
               temp2=temp2->link;
        }
        printf("Deleted data is %d",temp2->data);
        temp1->link=NULL;
        free(temp2);

}
```

Queue program

```
#include <stdio.h>
int que[10],size=10,front=0,rear=-1,val,n;
int main()
{
   //printf("1.enqueue\n2.dequeue\n3.display\n");
   while(1)
   {
      printf("1.enqueue\n2.dequeue\n3.display\n4.Exit");
      printf("Enter your choice \n");
      scanf("%d",&n);
```

```
        switch(n)
        {
            case 1:
            enqueue();
            break;
            case 2:
            dequeue();
            break;
            case 3:
            display();
            break;
            case 4:
            printf("Exiting....");
            exit(0);

        }
    }
}
enqueue()
{
    if(rear==size-1)
    {
        printf("Queue is full\n");
    }
    else
    {  rear++;
       printf("Enter a value to Add ");
       scanf("%d ",&val);
       que[rear]=val;
    }
}
```

```
dequeue()
{
   if(rear==-1)
   {
      printf("Queue is empty\n");
      return;
   }
   if(front>rear)
   {
      printf("No more elements to delete");
      return;
   }
   printf("Deleted element is %d",que[front]);
   front++;
}
display()
{
   for(int i=front;i<=rear;i++)
   {
      printf(" %d ",que[i]);
   }
}
```

Stack with linked list

```
#include <stdio.h>

struct node

{

int data;
```

```
struct node *link;

};

typedef struct node ll;

ll *top=NULL;

int data = 0, choice = 0, sizeoflist;

int

main ()

{

while (1)

{

printf("----Linked Stack----");

printf ("\nEnter your choice \n");

printf(" 1.Push \n 2.Pop \n 3.Print \n 4.Exit \n");
scanf ("%d", &choice);
switch (choice)

{

case 1: push(); break;

case 2: pop(); break ;

case 3: print(); break;

case 4: exit (   0);
```

```
default: printf ("Wrong choice");

}

}

}

void

menu (){

}

void push(){

ll *first;


     first=(ll *)malloc(sizeof(ll));

    printf("Enter Data ");

    scanf("%d",&data);


    first->data=data;
    first->link=top;
    top=first;
}

void pop()

{
ll *t1;
if(top == NULL)
```

```
{

printf("Stack Empty \n");

return;

}

t1=top;
printf("Deleted data is %d %u",t1->data,t1->link);

top=t1->link;
t1->link=NULL;
free(t1);


}

void print()
{
   ll *t2;
if(top == NULL)

   {

       printf("Stack Empty\n");

       return;

   }
   t2=top;
       while(t2!=NULL)

   {
        printf("%d %u\n",t2->data,t2->link);
        t2=t2->link;
```

```
    }
}
```

Queue With linked list

```
#include <stdio.h>

struct node

{

int data;

struct node *link;

};

typedef struct node ll;
ll *front=0,*rear=0;
int choice;
int main ()

{

while (1)

{

printf("----Linked Queue----");

printf ("\nEnter your choice \n");

printf(" 1.Enqueue \n 2.Dequeue\n 3.Print \n 4.Exit \n");
scanf ("%d", &choice);
switch (choice)
```

```
{

case 1: Enqueue(); break;

case 2: Dequeue(); break ;

case 3: print(); break;

case 4: exit (0);

default: printf ("Wrong choice");

}

}

}

void Enqueue(){
int data;
printf("Enter Data ");
scanf("%d",&data);
ll *first;
first=(ll *)malloc(sizeof(ll));
first->data=data;
first->link=NULL;
if(front==0 && rear==0)
{
   front=rear=first;
}
else
{
   rear->link=first;
   rear=first;
```

```
}
}
void Dequeue()
{
ll *t1;
if(front==0)
{
printf("Queue is Empty \n");
}
else
{
t1=front;
printf("Deleted data is %d %u",t1->data,t1->link);
front=front->link;
free(t1);
}
}

void print()
{
   ll *t2;
if(front==0 && rear==0)

   {

      printf("Stack Empty\n");

      return;

   }
   t2=front;
      while(t2!=NULL)

   {
       printf("%d %u\n",t2->data,t2->link);
```

```
        t2=t2->link;
    }
}
```

Copying elements from stack to queue:

```
//copying elements from stack to queue ;
#include <stdio.h>
int arr[5]={3,6,9,12,5},top=-1,size=5;
int stack[5];
int queue[5],front=0,rear=-1,i;
int main()
{
 push();
 print();
 enqueue();
 print1();

}
void push()
{
    if(top==size-1)
    {
        printf("Stack is full");
    }
    else
    {
        for(i=0;i<5;i++)
        {
            top++;
            stack[top]=arr[i];
        }
    }
}
```

```
void print()
{   printf("Stack elements are\n");
    if(top==-1)
    {
        printf("Stack is empty");
    }
    else
    {
        for(i=top;i>=0;i--)
        {
            printf("%d \n",stack[i]);
        }
    }
}
void enqueue()
{
    if(rear==size-1)
    {
        printf("queue is full");

    }
    else
    {
        for(i=0;i<5;i++)
        {
            rear++;
            queue[rear]=stack[top];
            top--;
        }

    }
}
void print1()
{   printf("Queue elements are\n");
    if(front==0 && rear==-1)
```

```
    {
        printf("queue is empty");
    }
    else
    {
        for(i=front;i<=rear;i++)
        {
            printf("%d  ",queue[i]);
        }
    }
}
```

Copying odd and even elements from linked list to the stack

```
#include <stdio.h>
int stack[10],stack1[10],top=-1,size=10,pos=0,i,top1=-1;
struct node
{
    int data;
    struct node *link;
};
 struct node *head=NULL;
 struct node *First;

int main()
{
    insertbeg(15);
    insertbeg(14);
    insertbeg(18);
    insertbeg(20);
    insertbeg(21);
    insertbeg(22);
    insertbeg(23);
    insertbeg(24);
```

```
    insertbeg(25);
    print1();
    push();
    print();
}
void insertbeg(int data)
{
    First=(struct node *)malloc(sizeof(struct node));
    First->data=data;
    First->link=head;
    head=First;
}
void push()
{
    struct node *tmp;
    tmp=head;
    while(tmp!=NULL)
    {
        pos++;
        if(pos%2!=0)
        {
            top++;
            stack[top]=tmp->data;
        }
        else
        {

            top1++;
            stack1[top1]=tmp->data;
        }
        tmp=tmp->link;
    }
}
void print()
{
```

```
   printf("Odd copied from list to stack are\n");
   if(top==-1)
   {
      printf("stack is empty");
   }
   else
   {
      for(i=top;i>=0;i--)
      {
         printf("%d \n",stack[i]);
      }
   }
   printf("Even copied from list to stack are\n");
   if(top1==-1)
   {
      printf("stack is empty");
   }
   else
   {
      for(i=top1;i>=0;i--)
      {
         printf("%d \n",stack1[i]);
      }
   }


}
void print1()
{
    printf("Elements in the list\n");
   if(head==NULL)
   {
      printf("list is empty");
   }
   else
```

```
  {
     struct node *tmp1;
     tmp1=head;
     while(tmp1!=NULL)
     {

        printf("%d \n",tmp1->data);
        tmp1=tmp1->link;
     }
  }
}
```

Program:

```
#include <stdio.h>

struct node

{

    int data;

    struct node *link;

};

struct node *head=NULL;

struct node *first;

int a[10][20];int r,c,i,j;

int a1[10]

int main()

{

    printf("enter no of rows:\n");

    scanf("%d",&r);
```

```
printf("enter no of columns:\n");
scanf("%d",&c);
for(i=0;i<r;i++)
{
    for(j=0;j<c;j++)
    {
        printf("enter array elements a[%d][%d]: ",i,j);
        scanf("%d",&a[i][j]);
    }
}
printf(" array elements:\n");
for(i=0;i<r;i++)
{
    for(j=0;j<c;j++)
    {
```

```
        printf("%d ",a[i][j]);
    }
    printf("\n");
}
for(i=0;i<r;i++)
{
    for(j=0;j<c;j++)
    {   if(i==0)
        {
            printf("%d ",a[i][j]);
        }
    }
}
firstrow();
//print();
```

```
}
void firstrow()
{
    first=(struct node *)malloc(sizeof(struct node));
    for(i=0;i<r;i++)
    {
        for(j=0;j<c;j++)
        {
            {
                first->data=a1[i][j];
                first->link=head;
                head=first;
            }
        }
    }
```

```
}
void print()
{
    if(head==NULL)
    {
        printf("List is empty");
    }
    else
    {
        struct node *tmp;
        tmp=head;
        while(tmp!=NULL)
        {
            printf("%d ",tmp->data);
            tmp=tmp->link;
```

```
        }

    }

}
```

FILES

Writing data into the file

```
#include
<stdio.h>
                    struct employee
                    {
                       int empID;
                       char empname[20];
                       char company[20];
                    }obj1[10];
                    int main()

                    {
                       int i;
                      for(i=0;i<3;i++)
                       {
                           printf("Enter employee ID ,employe name,employee
                    company\n");
                           scanf("%d %s
                    %s",&obj1[i].empID,&obj1[i].empname,&obj1[i].company
                    );

                       }
```

```
  /*for(int i=0;i<3;i++)
  {
      //printf("Enter employee ID ,employe
name,employee company\n");
      printf("%d %s
%s",obj1[i].empID,obj1[i].empname,obj1[i].company);

  }*/
  FILE *fp;
  fp=fopen("employeedb","wb");
  for(i=0;i<3;i++)
  {
     fwrite(&obj1[i],sizeof(obj1),1,fp);
  }
  fclose(fp);
}
```

Reading data into file:

```
#include
<stdio.h>
                struct employee
                {
                  int empID;
                  char empname[20];
                  char company[20];
                }obj1[10];
                struct employee obj2[10];
                int main()
                {
                  int i;
                  FILE *fp;
                  fp=fopen("employeedb","rb");
                  for(i=0;i<3;i++)
                  {
                     fread(&obj2[i],sizeof(obj2),1,fp);
                  }
                  for(i=0;i<3;i++)
```

```
    {
        printf("%d %s
  %s",obj2[i].empID,obj2[i].empname,obj2[i].company);
    }

    fclose(fp);
  }
```

Updating data into the file:

```
#include
<stdio.h>
                struct employee
                {
                  int empID;
                  char empname[20];
                  char company[20];
                }obj1[10];
                struct employee obj2[10];
                int main()
                {   int id;;int i;int index=0;
                  FILE *fp,*fp1;
                  fp=fopen("employeedb","r+b");
                  printf("enter employee id to update");
                  scanf("%d",&id);
                  while((fread(&obj2[i],sizeof(obj2),1,fp) ==1))
                  {
                     for(i=0;i<=3;i++)
                     {
                        if (obj2[i].empID == id)
                        {
                           printf("Enter the new data ");
                           scanf("%d %s %s",
                &obj2[i].empID,&obj2[i].empname,&obj2[i].company);
                           fseek(fp,sizeof(obj2[i])*i,SEEK_SET);
                           fwrite(&obj2[i], sizeof(obj2), 1, fp);
                        }
                     }
```

```
    }
    fclose(fp);
    printf("\nRecord updated.");
}
```

Deleting data in the file

```
#include
<stdio.h>
                struct employee
                {
                  int empID;
                  char empname[20];
                  char company[20];
                }obj1[10];
                struct employee obj2[10];
                int main()
                {    int id;;int i;int index=0;
                  FILE *fp,*fp1;
                  fp=fopen("employeedb","r+b");
                 fp1=fopen("employee","wb");
                  printf("enter employee id to update");
                  scanf("%d",&id);
                  while((fread(&obj2[i],sizeof(obj2),1,fp)
                ==1))
                  {
                     for(i=0;i<=3;i++)
                     {
                        if (obj2[i].empID!= id)
                        {
                           fwrite(&obj2[i], sizeof(obj2), 1,
                fp1);
                        }
                     }
                  }
                  fclose(fp);
                  fclose(fp1);
                  remove("employeedb");
```

```
   rename("employee","employeedb")
   printf("\nRecord deleted.");
}
```

Searching employee :

```
#include
<stdio.h>
                struct employee
                {
                   int empID;
                   char empname[20];
                   char company[20];
                }obj1[10];
                struct employee obj2[10];
                int main()
                {    int id;int flag;int i;int index=0;
                   FILE *fp;
                   fp=fopen("employeedb","rb");
                   printf("enter employee id to search");
                   scanf("%d",&id);
                while((fread(&obj2[i],sizeof(obj2),1,fp) ==1))

                {
                   for(i=0;i<=3;i++)
                   {
                      if(obj2[i].empID==id)
                      {
                         flag=1;

                         printf("found");
                         printf("%s
%s",obj2[i].empname,obj2[i].company);
                         break;
                      }
                   }
                }
```

```
  /* if(flag==1)
   {
       printf("found");
       printf("%s
 %s",obj2[index].empname,obj2[index].company);
   }*/
 if(flag==0)
   {
    printf("not found");
   }
  fclose(fp);
 }
```

Write a c program to create 2 dimensional array of integers with say 4 rows 3 cols. Now from this 2 d array create a 4(rows) linked list with elements from each row to fit into linked list

```
#include<stdi
o.h>
                #include<stdlib.h>
                int a[4][3],i,j,b[10],k=0;
                struct node
                {
                int data;
                struct node *nex;
                };
                struct node* head[10];


                void d2()
```

```
{
for(i=0;i<4;i++)
{
for(j=0;j<3;j++)
scanf("%d",&a[i][j]);
}
for(i=0;i<4;i++)
{
for(j=0;j<3;j++)
printf("%d\t",a[i][j]);
printf("\n");
}
}


void ha()
{
for(i=0;i<4;i++)
{
struct node* new=(struct
node*)malloc(sizeof(struct node*));
new->data=a[i][0];
new->nex=NULL;
head[i]=new;


for(j=1;j<3;j++)
{
struct node* new=(struct
node*)malloc(sizeof(struct node*));
new->data=a[i][j];
new->nex=NULL;
struct node* p;
p=head[i];
while(p->nex!=NULL)
{
p=p->nex;
}
p->nex=new;
}
```

```
}
}


void dis()
{
struct node* p1;
for(i=0;i<4;i++)
{
printf("%u->",head+i);
p1=head[i];
while(p1!=NULL)
{
printf("%d\t",p1->data);
p1=p1->nex;
}
printf("\n");
}
}


int main()
{int c;
while(1)
{
scanf("%d",&c);
switch(c)
{
case 1:d2();
        break;
case 2:ha();
        break;
case 3:dis();
        break;
default:exit(0);
}
}
return 0;
}
```