Array of Structures 1

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
struct student
{
   char name[10];
   int roll, year;
}s[2]={{"Michael",7,87},{"Preetam",12,18}};
int main()
{
   int i;
  // for (i=0;i<20;i++)
   //strcpy(s[i].name,"Michael");
   printf("%s",s[1].name);
}
```

Array of Structures (Method 2)

```
#include<stdio.h>
#include<stdlib.h>
struct student
{
   int roll, m1;
   char name[30];
   }s[20];
int main ()
{
   int i, n;
   printf("Enter the number of students: ");
   scanf("%d", &n);
   for (i=1; i<=n; i++)
   {
   printf("Enter details of student%d: ", i);
      scanf("%d %d %s", &s[i].roll, &s[i].m1, s[i].name);
}
for (i=1; i<=n; i++)
   {
   printf("details of student%d are %d %d %s\n", i, s[i].roll, s[i].m1, s[i].name);
   }
}
```

Nested Structures 1

```
#include<stdio.h>
#include<stdlib.h>
struct dob
{
int dd, mm, yy;
}date;
struct student
{
   int roll, m1;
   char name[30];
   struct dob date;
}s1= {7, 98, "Michael", {12, 10, 87}};
int main ()
{
   printf("%d %d %s %d %d %d", s1.roll, s1.m1, s1.name, s1.date.dd, s1.date.mm, s1.date.yy);
   return 0;
}
```

Nested Structures (Method 2)

```
#include<stdio.h>
#include<stdlib.h>
struct dob
{
   int dd, mm, yy;
};
struct stu
```

```
{
   char name[20];
   int roll;
   struct dob date;
}s1;
int main()
{
   printf("Enter the name: ");
   scanf("%s",s1.name);
   s1.roll = 7;
   printf("DOB in dd mm yy:");
   scanf("%d %d %d", &s1.date.dd, &s1.date.mm, &s1.date.yy);
   printf("%s %d %d-%d-%d", s1.name, s1.roll, s1.date.dd, s1.date.mm, s1.date.yy);
}
```

Nested Structures (Method 3)

```
#include<stdio.h>
#include<stdlib.h>
struct stu
{
   char name[20];
   int roll;
  struct dob
{
   int dd, mm, yy;
}date;
}s1 = {"Michael", 7, {12, 10, 87}};
int main()
{
      printf("%s %d %d-%d-%d", s1.name, s1.roll, s1.date.dd, s1.date.mm, s1.date.yy);
}
```

Nested Structures (Method 4)

```
#include<stdio.h>

#include<stdlib.h>

struct DOB

{

   int dd, mm, yy;

}date = {77,77,77};

struct student

{

   int roll;

   char name[30];

   struct DOB date;

}s1 = {7, "Mich", {12, 10, 87}};

int main()

{

printf("Mr. %s with roll number %d was born on %d-%d-%d", s1.name, s1.roll, s1.date.dd, date.mm,
s1.date.yy);//Please observe date.mm and s1.date.mm

}
```

Nested Structures (Method 5)

```
#include<stdio.h>

#include<string.h>

struct DOB

{

   int dd, mm, yy;

}date = {77,77,77};

struct student

{

   int roll;

   char name[30];

   struct DOB date;

}s1 = {7, "Mich", {12, 10, 87}};

int main()
```

```
{
strcpy(s1.name, "pree");
printf("Mr. %s with roll number %d was born on %d-%d-%d", s1.name, s1.roll, s1.date.dd, date.mm,
s1.date.yy);
}
```

1. Write a C program to store name, roll number, year and marks of three subjects of n students and print the student the name, roll number, average and grade based on average marks of the student using structures.

```
# include<stdio.h>
# include<stdlib.h>
struct student{
   char name[30], grade;
   int roll, yearr, m1, m2, m3;
   float avgg;
}s[30];
int main()
{
   int i, n;
   printf("Enter the number of students: ");
   scanf("%d", &n);
   for (i=1;i<=n;i++)
   {
   printf("Enter the details of student%d: ",i);
   scanf("%s %d %d %d %d %d", s[i].name, &s[i].roll, &s[i].yearr, &s[i].m1, &s[i].m2, &s[i].m3);
   s[i].avgg = (float)((s[i].m1 + s[i].m2 + s[i].m3)/3);
   if (s[i].avgg >= 75)
   s[i].grade = 'A';
   else if (s[i].avgg >= 50)
   s[i].grade = 'B';
   else
   s[i].grade = 'C';
   }
    for (i=1;i<=n;i++)
   {
```

```
   printf("The details of student%d are name = %s, roll = %d, average = %f and grade = %c\n", i, s[i].name,
s[i].roll, s[i].avgg, s[i].grade);

   }

}
```

<u>Name and Roll method 1</u>

```
#include<stdio.h>

#include<stdlib.h>

   int main()

{

   char name[7]="Michael";

   int roll, year;

   roll=7;

   printf("%s %d",name,roll);

   return 0;

   }
```

<u>Output:</u>

```
Michael 7
```

<u>Name and Roll method 2</u>

```
#include<stdio.h>

#include<stdlib.h>

   int main()

{

   char name[7]={'M','i','c','h','a','e','l'};

   int roll, year;

   roll=7;

   printf("%s %d",name,roll);

   return 0;

   }
```

<u>Output:</u>

```
Michael 7
```

<u>Name and Roll method 3</u>

```
#include<stdio.h>

#include<stdlib.h>
```

```
#include<string.h>
struct stu
{
   char name[7];
   int roll;
}s1;
int main()
{
   strcpy(s1.name,"Michael");
   s1.roll=7;
   printf("%s %d",s1.name,s1.roll);
   return 0;
   }
```

Output:

```
Michael 7
```

Write a C program to read employee details employee number, employee name, basic salary of an employee using structures and create a pointer variable for the employee and print employee number, employee name and gross salary using pointer variable.

```
#include<stdio.h>
#include<stdlib.h>
struct emp
{
int num, bs, hra, da, pt,gs;
char name[30];
}*e1;
int main()
{
e1=malloc(sizeof(struct emp));
printf("Enter employee name, number and basic salary: ");
scanf("%s%d%d",e1->name,&e1->num,&e1->bs);
e1->hra=0.15*e1->bs;
e1->da=0.05*e1->bs;
e1->pt=0.10*e1->bs;
e1->gs=e1->bs+e1->hra+e1->da-e1->pt;
printf("The employee %s with number %d gets a gross salary of %d",e1->name,e1->num,e1->gs);
return 0;
}
```

Output:

```
Enter employee name, number and basic salary: Michael
7
90000
The employee Michael with number 7 gets a gross salary of 99000
Process returned 0
```

Write a C program that uses functions to perform the following:

a. Create a node structure consists of data (integer) and address

b. Insert the elements one after the another in a sequence

c. Display the contents of the above list.

```
#include<stdio.h>
#include<stdlib.h>
int i,n;
struct node
{
int data;
struct node *next;
}s[5];
void insert()
{
printf("Enter the number of elements:");
scanf("%d",&n);
printf("Enter the data elements");
for(i=0;i<n;i++)
scanf("%d",&s[i].data);
}
void display()
{
```

```
printf("The contents of the list are: ");

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

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

}

int main()

{

insert();

display();

return 0;

}
```

Output:

```
Enter the number of elements:3

Enter the data elements7

8

9

The contents of the list are: 7 8       9

Process returned 0 (0x0)
```

student_details_pointer_

```
#include<stdio.h>

#include<stdlib.h>

struct node

{

char n[30];

struct node *next;

}s[7];

int main()

{

   int i,n;

printf("Enter no. of els:");

scanf("%d",&n);

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

{
```

```
printf("Enter el%d",i);

scanf("%s",s[i].n);

}

printf("%s",s[0]);

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

printf("->%s",s[i]);

}
```

STACKS

Implementation of Stack using Arrays

```
#include<stdio.h>
#include<stdlib.h>
#include<stdbool.h>
# define N 5
int stac[N], top=-1, i;
char ch;
bool isfull()
{
   if (top>=N-1)
   return true;
   else
   return false;
}
bool isempty()
{
   if (top<=-1)
   return true;
   else
   return false;
}
void push()
{
   if(isfull())
   printf("\nStack overflow");
   else
   {
   printf("\nEnter the element:");
   top++;
   scanf("%d",&stac[top]);
}
}
void pop()
{
   if(isempty())
   printf("Stack underflow");
   else
   {
   printf("\nThe popped element is %d",stac[top]);
   top--;
}
}
void display()
{
   if(top>=0)
   {
      printf("\nThe Stack elements are:");
      for(i=top;i>=0;i--)
      printf("%d\t",stac[i]);
   }
}
int main()
{
   do{
   printf("Enter a to add, r to remove, d to display elements of stack. Enter e to exit:");
   scanf("%s",&ch);
      switch(ch)
   {
      case 'a':push();
      break;
      case 'r':pop();
      break;
      case 'd':display();
      break;
      case 'e': break;
      default:printf("Enter valid input");
      break;
   }
   }while(ch!='e');
   }
```

<u>Implementation of Stack using Linked Lists</u>

```
#include<stdio.h>

#include<stdlib.h>

struct node

{

int data;

struct node *next;

}*st, *tp, *nn;

void display()

{

tp=st;

printf("\nThe linked list is: ");

while(tp!=NULL)
```

```
{
printf("%d\t",tp->data);
tp=tp->next;
}
}
void insert_at_beg()
{
   nn=malloc(sizeof(struct node));
   printf("\nEnter the node you wish to insert at beginning:");
   scanf("%d",&nn->data);
   nn->next=st;
   st=nn;
   }
delete_at_beg()
{
   tp=st;
   st=st->next;
   free(tp);
}
int main()
{
   int ch;
//create();
do{
printf("\nEnter 1 to push, 2 to pop, 3 to display, 0 to exit:");
scanf("%d",&ch);
switch(ch)
{
   case 0: break;
   case 1: insert_at_beg();
   break;
   case 2: delete_at_beg();
   break;
```

```
    case 3: display();
    break;
    default: printf("\nInvalid input:");
     break;
}
}while(ch!=0);
return 0;
}
```

<u>Infix to postfix conversion</u>

```
    #include<stdlib.h>
    #include<stdio.h>
    #include<string.h>
    #include<ctype.h>
    struct node
    {
    char data;
    struct node *next;
    }*top=NULL,*tp,*nn;
    void push(char x)
    {
        nn=malloc(sizeof(struct node));
        nn->data = x;
        nn->next = top;
        top=nn;
    }
    void pop()
    {
        if(((top->data)!='(')&&((top->data)!=')'))
        printf("%c", top->data);
        tp=top;
        top=top->next;
        free(tp);
    }
```

```
int priority(char x)

{

    if(x == '(')

        return -1;

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

        return 1;

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

        return 2;

    if(x == '^')

        return 3;

}

int main()

{

    int i,l;

    char exp[100];

    printf("Enter the expression : \n");

    scanf("%s",exp);

    l=strlen(exp);

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

    {

        if(isalnum(exp[i])) //The function isalnum() is used to check that the character is
alphanumeric or not. It returns non-zero value, if the character is alphanumeric (means letter
or number) otherwise returns zero. It is declared in “ctype.

            printf("%c",exp[i]);

        else if(exp[i] == '(')

            push(exp[i]);

        else if(exp[i] == ')')

        {

            while(top->data!= '(')

                pop();

                pop();

        }

        else

        {
            while(top!=NULL && priority(top->data) >= priority(exp[i]))
                pop();
            push(exp[i]);
        }
    }
    while(top != NULL)
        pop();
return 0;
}
```

<u>Evaluation of postfix expression</u>

```
#include<stdio.h>
#include<stdlib.h>
void add();
void sub();
void mult();
void division();
void rem();
void power();
int stack[50],top=-1;
int main()
{
    char st[50];
    int i;
    printf("enter the postfix expression : ");
    scanf("%s",st);
    for(i=0;st[i]!='\0';i++)
    {
        switch(st[i])
        {
            case '+':add();
                     break;
            case '-':sub();
                     break;
            case '*':mult();
                     break;
            case '/':division();
                     break;
            case '%':rem();
                     break;
            case '^':power();
                    break;
            default: top++;
                    stack[top]=st[i] - 48;
        }
    }
    printf("The result of postfix expression is = %d ",stack[top]);
}
void add()
{
    int op1,op2,res;
    op1=stack[top];
    top--;
    op2=stack[top];
    top--;
    res=op2+op1;
    top++;
    stack[top]=res;
}
void sub()
{
    int op1,op2,res;
    op1=stack[top];
    top--;
    op2=stack[top];
    top--;
    res=op2-op1;
```

```
    top++;

    stack[top]=res;

}

void mult()

{

    int op1,op2,res;

    op1=stack[top];

    top--;

    op2=stack[top];

    top--;

    res=op2*op1;

    top++;

    stack[top]=res;

}

void division()

{

    int op1,op2,res;

    op1=stack[top];

    top--;

    op2=stack[top];

    top--;

    res=op2/op1;

    top++;

    stack[top]=res;

}

void rem()

{

    int op1,op2,res;

    op1=stack[top];

    top--;

    op2=stack[top];

    top--;

    res=op2%op1;
    top++;
    stack[top]=res;
}
void power()
{
    int op1,op2,res=1,i;
    op1=stack[top];
    top--;
    op2=stack[top];
    top--;
    for(i=0;i<op1;i++) // for repetitive multiplication
        res=res*op2;
    top++;
    stack[top]=res;
}
```

stack_implementation_using_array_ part1

```
#include<stdio.h>
#include<stdlib.h>
#include<stdbool.h>
# define N 50
char stac[N];
int i, top=-1;
bool isfull()
{
    if(top>=N-1)
        return true;
    else
        return false;
}
bool isempty()
{
    if(top<=-1)
        return true;
    else
        return false;
}
void push()
{
    if(isfull())
        printf("overflow condition");
    else
    {
        top++;
        printf("Enter element to push: ");
        scanf("%s",&stac[top]);
    }
}
void pop()
{
    if(isempty())
        printf("underflow condition");
    else
    {
        printf("The popped element is: %c\n",stac[top]);
        top--;
    }
}
void display()
{
    printf("The stack elements are: ");
    for(i=top;i>-1;i--)
    printf("%c\n",stac[i]);
}
void del_3()
{
    pop();
    pop();
    pop();
}
int main()
{
    char ch;
    do{
        printf("Enter a to add; Enter r to remove; d to display; t to delete three elements; e to exit: ");
        scanf("%s",&ch);
        switch(ch)
        {
            case 'a': push();
            break;
            case 'r': pop();
            break;
            case 'd': display();
            break;
            case 't': del_3();
            break;
            case 'e': break;
            default: printf("Enter valid input: ");
            break;
        }
    } while(ch!='e');
return 0;
}
```

stack_implementation_using_array_palindrome_part2_simplified

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#define N 50
char stack[N], ins[N], outs[N];
```

```
int top=-1;
void push(int x)
{
        top++;
      stack[top]=x;
}
char pop()
{
        top--;
        return stack[top+1];
}
int main()
{
    int i,s;
    printf("Enter input string:");
    scanf("%s",ins);
    s=strlen(ins);
    for(i=0;i<s;i++)
    push(ins[i]);
    for(i=0;i<s;i++)
    outs[i]=pop();
    printf("The input string is %s",ins);
    printf("\nThe output string is %s",outs);
    if(strcmp(ins,outs))
    printf("\nit is not a palindrome");
    else
                printf("\nit is palidrome");
    return 0;
}
```

stack_implementation_using_array_reverse_a_string_part3

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

```
#include<string.h>

#include<stdbool.h>

# define N 50

char stac[N], ins[N], outs[N], count=0;;

int i,j=0,re=0,top=-1;

bool isfull()

{

    if(top>=N-1)

        return true;

    else

        return false;

}

bool isempty()

{

    if(top<=-1)

        return true;

    else

        return false;

}

void push(char x)

{

    if(isfull())

        printf("overflow condition");

    else

    {

        top++;

        stac[top] = x;

            }

}

void pop()

{

       if(isempty())

        printf("underflow condition");
    else
    {
        printf("The popped element is: %c\n",stac[top]);
         outs[j]=stac[top];
        top--;
        j++;
    }
}
void display()
{
    printf("The stack elements are: ");
    for(i=top;i>-1;i--)
    printf("%c\n",stac[i]);
}
int main()
{
    int l;
    printf("Enter input string: ");
    scanf("%s",ins);
    l=strlen(ins);
    for(i=0;i<l;i++)
        push(ins[i]);
        for(i=0;i<l;i++)
        pop();
        printf("Input string is %s\n",ins);
    printf("Output string is %s\n",outs);
    for(i=0;i<l;i++)
    {
    if (ins[i]==outs[i])
    re++;
    }
    printf("string length is %d and re value is %d\n",l,re);
    if(re==l)
```

```
        printf("Palindrome");
    else
        printf("Not a Palindrome");
return 0;
}
```

stack_implementation_using_stack_all_in_one _part4

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
    char data;
    struct node *next;
}*top=NULL,*tp,*nn;
void push()
{
    nn=malloc(sizeof(struct node));
    printf("Enter element to push: ");
    scanf("%s",&nn->data);
    nn->next=top;
    top=nn;
}
void pop()
{
    printf("The popped element is: %c\n",top->data);
    tp=top;
    top=top->next;
    free(tp);
}
void display()
{
    tp=top;
        printf("The stack elements are: ");
    while(tp!=NULL)
    {
    printf("%c\t",tp->data);
    tp=tp->next;
}
}
void del_3()
{
 pop();
 pop();
 pop();
}
int main()
{
    char ch;
    do{
        printf("Enter a to add; Enter r to remove; d to display; t to delete three elements; e to exit: ");
        scanf("%s",&ch);
        switch(ch)
        {
            case 'a': push();
            break;
            case 'r': pop();
            break;
            case 'd': display();
            break;
            case 't': del_3();
            break;
            case 'e': break;
            default: printf("Enter valid input: ");
            break;
        }
    } while(ch!='e');
return 0;
```

```
}
```

Student Assignment: Balanced Delimiters

QUEUES

queue_using_array

```
#include<stdio.h>
#include<stdlib.h>
#include<stdbool.h>
# define N 5
int Queu[N],front=-1,rear=-1;
bool isfull()
{
   if(rear>=N-1)
   return true;
   else
   return false;
}
bool isempty()
{
      if((rear<=-1)&&(front<=-1))
   return true;
   else
   return false;
}
void enqueue()
{
   if(isfull())
  printf("Overflow condition");
  else
  {
     if(isempty())
     front++;
```

```
       rear++;
       printf("Enter data:");
       scanf("%d",&Queu[rear]);
    }
}
void dequeue()
{
    if(isempty())
   printf("Underflow condition");
   else
   {
    printf("The dequeued element is %d\n",Queu[front]);
    front++;
    if(front>rear)
    front=rear=-1;
}
}
void display()
{
    int i;
    printf("The elements in queue are: ");
    for(i=front;i<=rear;i++)
     printf("%d\t",Queu[i]);
}
int main()
{
    char ch;
    do{
    printf("Enter a to add, r to remove, d to display and e to exit: ");
    scanf("%s",&ch);
    switch(ch)
   {
      case 'a': enqueue();
      break;
            case 'r': dequeue();
      break;
            case 'd': display();
      break;
            case 'e':      break;
            default:printf("Enter valid input: ");
            break;
   }
   }while(ch!='e');
   return 0;
}
```

<u>queue_using_array (Minimized; without using boolean function)</u>

```
#include<stdio.h>
#include<stdlib.h>
# define N 5
int Queu[N],front=-1,rear=-1;
void enqueue()
{
   if(rear==N-1)
  printf("Overflow condition");
  else
  {
     if((rear==-1)&&(front==-1))
     front++;
     rear++;
     printf("Enter data:");
     scanf("%d",&Queu[rear]);
  }
```

```
}
void dequeue()
{
    if((rear==-1)&&(front==-1))
   printf("Underflow condition");
   else
   {
    printf("The dequeued element is %d\n",Queu[front]);
    front++;
    if(front>rear)
    front=rear=-1;
}
}
void display()
{
    int i;
    printf("The elements in queue are: ");
    for(i=front;i<=rear;i++)
     printf("%d\t",Queu[i]);
}
int main()
{
    char ch;
    do{
    printf("Enter a to add, r to remove, d to display and e to exit: ");
    scanf("%s",&ch);
    switch(ch)
    {
        case 'a': enqueue();
        break;
                case 'r': dequeue();
        break;
            case 'd': display();
        break;
            case 'e':      break;
            default:printf("Enter valid input: ");
            break;
    }
    }while(ch!='e');
    return 0;
}
```

queue_using_SLL

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
    int data;
    struct node *next;
}*st,*nn,*tail;
void enqueue()
{
    nn=malloc(sizeof(struct node));
    printf("Enter data: ");
    scanf("%d",&nn->data);
    if(st==NULL)
    {
        st=nn;
        tail=nn;
    }
    else
    {
        tail->next=nn;
        tail=tail->next;
        tail->next=NULL;
    }
}
void dequeue()
{
    nn=st;
    st=st->next;
    free(nn);
}
void display()
{
    nn=st;
    printf("The elements in queue are:");
    while(nn!=NULL)
    {
        printf("%d\t",nn->data);
        nn=nn->next;
    }
}
int main()
{
    char ch;
    do{
    printf("Enter a to add, r to remove, d to display and e to exit: ");
    scanf("%s",&ch);
    switch(ch)
    {
        case 'a': enqueue();
        break;
```

```
            case 'r': dequeue();
        break;
            case 'd': display();
        break;
            case 'e':     break;
            default:printf("Enter valid input: ");
            break;
    }
    }while(ch!='e');
    return 0;
}
```

circular_queue_using_array

```
#include<stdio.h>
#include<stdlib.h>
#include<stdbool.h>
# define N 5
int Queu[N],front=-1,rear=-1;
bool isfull()
{
    if(rear==(front+N-1)%N)
    return true;
    else
    return false;
}
bool isempty()
{
        if((rear==-1)&&(front==-1))
    return true;
    else
    return false;
```

```
}
void enqueue()
{
   if(isfull())
  printf("Overflow condition");
  else
  {
     if(isempty())
     front=0;
     rear=(rear+1)%N;
     printf("Enter data:");
     scanf("%d",&Queu[rear]);
  }
}
void dequeue()
{
   if(isempty())
  printf("Underflow condition");
  else
  {
   printf("The dequeued element is %d\n",Queu[front]);
   front=(front+1)%N;
   if(front==(rear+1)%N)
   front=rear=-1;
}
}
void display()
{
   int i;
   printf("The elements in queue are: ");
   i=front;
   do{
   printf("%d\t",Queu[i]);
      i=(i+1)%N;
   }while(i!=(rear+1)%N);
}
int main()
{
   char ch;
   do{
   printf("Enter a to add, r to remove, d to display and e to exit: ");
   scanf("%s",&ch);
   switch(ch)
   {
      case 'a': enqueue();
      break;
            case 'r': dequeue();
      break;
            case 'd': display();
      break;
            case 'e':       break;
            default:printf("Enter valid input: ");
            break;
   }
   }while(ch!='e');
   return 0;
}
```

<u>Circular Queue using linked list:</u>

```
#include<stdio.h>//Mostly same as Queue using linked list. Non-common highlighted in yellow
#include<stdlib.h>
struct node
```

```
{
   int data;
   struct node *next;
}*st,*nn,*tail;
void enqueue()
{
   nn=malloc(sizeof(struct node));
   printf("Enter data: ");
   scanf("%d",&nn->data);
   if(st==NULL)
   {
      st=nn;
      tail=nn;
   }
   else
   {
      tail->next=nn;
      tail=tail->next;
      tail->next=NULL;
   }
   tail->next=st;
}
void dequeue()
{
   nn=st;
   st=st->next;
   free(nn);
   tail->next=st;
}
void display()
{
   nn=st;
   printf("The elements in queue are:");
   do{
      printf("%d\t",nn->data);
      nn=nn->next;
   }while(nn!=st);
}
int main()
{
   char ch;
   do{
   printf("Enter a to add, r to remove, d to display and e to exit: ");
   scanf("%s",&ch);
   switch(ch)
   {
      case 'a': enqueue();
      break;
            case 'r': dequeue();
      break;
            case 'd': display();
      break;
            case 'e':      break;
            default:printf("Enter valid input: ");
            break;
   }
   }while(ch!='e');
   return 0;
}
```

Search elements in Circular Queue using linked list:

#include<stdio.h> //Mostly same as **Circular Queue using linked list. Non-common in pink**

```
#include<stdlib.h>
struct node
{
   int data;
   struct node *next;
}*st,*nn,*tail;
void enqueue()
{
   nn=malloc(sizeof(struct node));
   printf("Enter data: ");
   scanf("%d",&nn->data);
   if(st==NULL)
   {
      st=nn;
      tail=nn;
   }
   else
   {
      tail->next=nn;
      tail=tail->next;
      tail->next=NULL;
   }
   tail->next=st;
}
void dequeue()
{
   nn=st;
   st=st->next;
   free(nn);
   tail->next=st;
}
```

```
void searchh()
{
  int sr;
  nn=st;
  printf("Enter search element:");
  scanf("%d",&sr);
  do{
    if(nn->data==sr)
    {
    printf("Present");
    exit(0);
    }
    nn=nn->next;
  }while(nn!=st);
  printf("Not Present\n");
}
int main()
{
  char ch;
  do{
  printf("Enter a to add, r to remove, s to search, and e to exit: ");
  scanf("%s",&ch);
  switch(ch)
  {
    case 'a': enqueue();
    break;
        case 'r': dequeue();
    break;
        case 's': searchh();
    break;
        case 'e':     break;
```

```
            default:printf("Enter valid input: ");
            break;
    }
    }while(ch!='e');
    return 0;
}
```

Priority Queue using array

```
#include<stdio.h>
int pq[3][5], v, p, f[3]={-1,-1,-1}, r[3]={-1,-1,-1},t;
void enque()
{
printf("Enter val, priority: ");
scanf("%d%d", &v, &p);
if((f[p]==-1) && (r[p]==-1))
f[p]=0;
r[p]=r[p]+1;
t=r[p];
pq[p][t]=v;
}
void deque()
{
    p=0;
    while(f[p]==-1 && r[p]==-1)
        p++;
    f[p]=f[p]+1;
    if(f[p]>r[p])
    {
f[p]=r[p]=-1;
    }
}
void disp()
```

```
{
int i;
p=0;
while(p<3)
{
   if((f[p]==-1) && (r[p]==-1));
       else{
   for (i=f[p];i<=r[p];i++)
printf("%d    ",pq[p][i]);}
p++;
}
}
int main()
{
int ch;
do{
printf("Enter 0 to enque,1 to deque, 2 to dsip, 3 to exit: ");
scanf("%d",&ch);
switch(ch)
{
case 0: enque();
break;
case 1: deque();
break;
case 2: disp();
break;
case 3:
break;}
}while(ch!=3);
return 0;
}
```