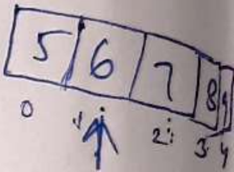
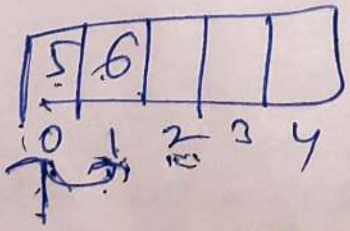
Priority Queue :-

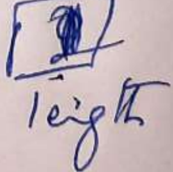
```
#define max 5
int q[max], front = -1, rear = -1;
void insertion ( )
{
    int el;
    if (rear == max-1)
    {
        printf (" full");
    }
    else
    {
        printf (" enter element");
        scanf (" %d", &el);
        if (front == -1)
        {
            front = rear = 0;
            q[~~front~~ rear] = el;
        }
        else
        {       int i, j;
            for (i = front ; i <= rear ; i++)
            {
                if (q[i] > el)
                    break;
            }
            for (j = rear ; j >= i ; j--)
                    q[j+1] = q[j];
            q[i] = el;
            rear++;
}   }   }
```

```
void deletion ( )
{
    int k;
    ~~if( rear == n~~
    if (front == -1)
    {
        printf(" empty");
    }
    ~~else~~ ~~k = q[front];~~
        ~~front++;~~
        ~~printf(" deleted element is~~
    else
    {
        k = q[front];
        if(front == rear)
            front = rear = -1;
        else
            front ++;
        printf(" deleted element is %d", k);
    }
}
void display ( )
{
  int i;
    for (i= front ; i<=rear ; i++)
        printf(" %d", q[i]);
}
main( )
{
   int ch;
   while (1)
   {
       printf(" enter 1 to insert 2 to delete
                 3 to display 4 to exit");
       scanf(" %d", &ch);
switch(ch)
{
    case 1: insertion ( ); break;
    case 2: deletion ( ); break;
    case 3; display( ); break;
    case 4; exit(0);
  }
 }
}
```

```
#define size ~~max~~ 5
int l[size ~~max~~], length =0;
void insertion ()
{
    int index, el;
    printf(" enter index");
    scanf(" %d", &index);
    if( length == size)
    {
        printf(" full");
    }
    ~~else if ( length < 0 && length > si~~
    else if (index < 0 || index > length)
    {
        printf(" insertion not possible");
    }
    else
    {
        printf(" enter element");
        scanf(" %d", &el);
        for (i = length-1 ; i >= index ; i--)
        {
                ~~if~~ l[i+1] = l[i];
        }
          l[index] = el;
          length ++;
    }
}
```







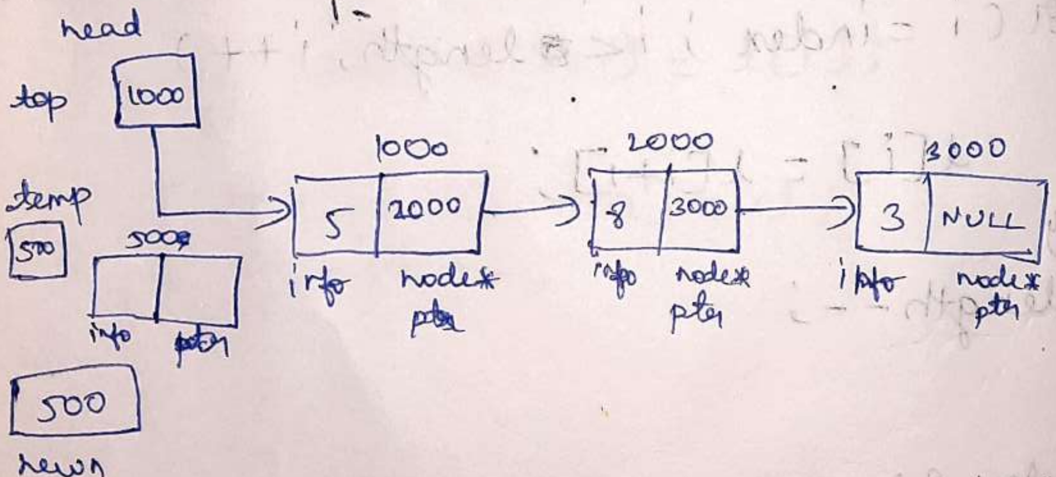
i>=0 .   1>0

```
void deletion ( )
{  int index; printf(" enter index"); scanf(" %d", &index);
  if ( length = = 0 )
  {
      printf(" empty");
  }                         ||
  else if ( index < 0  ~~&&~~  index >= length )
  {
     printf(" not possible");
  }
  ~~else~~
  {       ~~printf(" deleted element is %d", l[index]~~
  ~~for ( i = index ; i <= length - 1 ; i++ )~~
     {
          ~~l[index] printf(" deleted is %d", l[index]~~
          ~~l[i] = l[i+1];~~
  else

  {   ~~printf(" enter index");~~
      ~~scanf~~
      printf(" deleted element is %d", l[index]);
      for ( i = index ; i < length-1 ; i++ )
      {
          l[i] = l[i+1];
      }
      length--;
  }
}
void display ( )
{
  int i;
  for ( i = 0 ; i < length ; i++ )
        printf(" %d ", l[i]);
}
```

```
main()
{
    int ch;
    while(1)
    {
        printf("enter 1 to insert 2 to delete
                3 to display 4 to exit");
        scanf("%d", &ch);
        switch(ch)
        {
            case 1: insertion(); break;
            case 2: deletion(); break;
            case 3: display(); break;
            case 4: exit(0);
        }
    }
}
```

<u>stack using linked list</u>:-



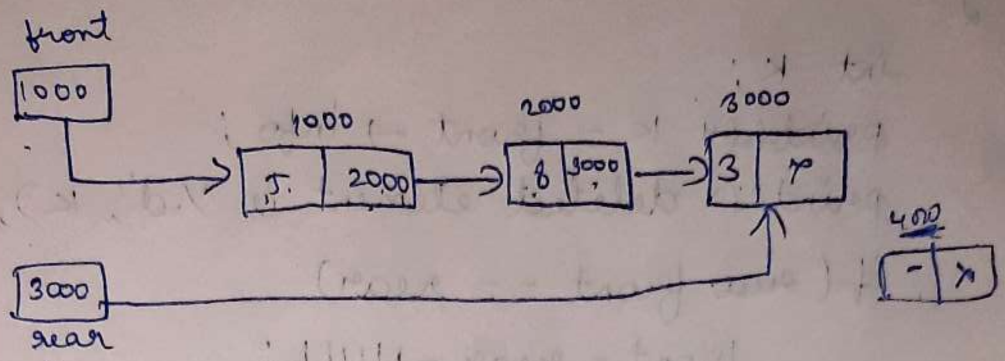
<u>Linked List</u>:-

It is logical linear data structure

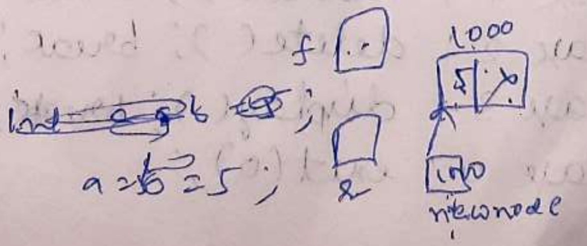
```
struct node        (Self referential structure).
 {
   int ~~ifo~~ info;
    struct node *ptr;
  };
typedef struct node node;
node *top = NULL, *temp , *newnode;
void push()
{
  int ele;
  printf(" enter element ");
   scanf(" %d", &ele);
  newnode = (node *) malloc(sizeof(node));
    newnode -> info = ele;
    newnode -> ptr = top;
      top = newnode;
}
void pop()
{
  int k;
  if(top == NULL)
  {
    printf(" empty ");
  }
  else
  {
     k = top -> info;
     top = top -> ptr;
     printf(" deleted element is %d", k);
  }
}
```

```
void display ( )
{
    temp = top;
    while (temp != NULL)
    {
        printf(" %d", temp -> info);
        temp = temp -> ptr;
    }
}
main ( )
{
    int ch;
    while (1)
    {
        printf(" enter 1 to push 2 to pop
                 3 to display 4 to exit ");
        scanf(" %d", &ch);
        switch (ch)
        {
            case 1: push( ); break;
            case 2: pop ( ); break;
            case 3: display ( ); break;
            case 4: exit (0);
        }
    }
}
```

queue using linked list :-



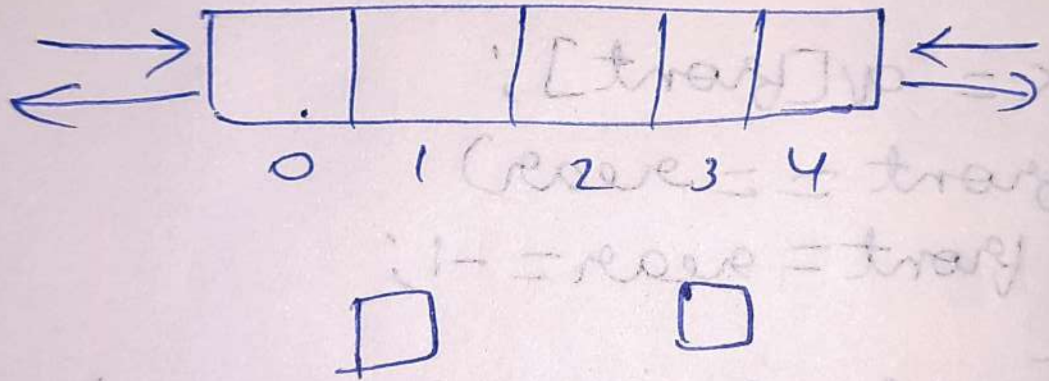
```
struct node
{
    int info;
    struct node *ptr;
};
~~type def~~                    node;
typedef  struct node  ~~*front = NULL, *rear~~
                        ~~= NULL~~
node ~~&~~ *front = ~~*~~rear = NULL, *temp, *newnode;
void insert ( )
{   int el;
    newnode = (node *) malloc ( sizeof (node));
    printf("enter element");
    scanf("%d", &el);
    newnode -> info = el;
    newnode -> ptr = NULL;
    if (front == NULL)
        front = rear = newnode;
    else
        rear = rear -> ptr = newnode;
}
```



```
void delete()
{
    int k;
    ~~printf(~~ k = front -> info;
    printf("deleted element is %d", k);
    if( front == rear)
            front = rear = NULL;

    else
          front = front -> ptr;
 }
void display()
{
        temp = front;
      while (temp != NULL)
      {
              printf("%d", temp -> info);
 }   }         temp = temp -> ptr;
void main()
{
    int ch;
    while(1)
    {
      printf("enter 1 to insert 2 to delete
                  3 to display  4 to exit");
     scanf("%d", &ch);
     switch(ch)
     {
           case 1 : insert(); break;
           case 2 : delete(); break;
           case 3 : display(); break;
           case 4 : exit(0);
     }
   }
}
```

Double ended queue:-



```
#include <stdio.h>
 int   q[max], front =-1, rear=-1;
void  insertf ( )
{   int el;
   if(front ==0 && rear==max-1)
   {
      printf(" full");
   }
   else
   {
      printf(" enter element");
       scanf(" %d", &el);
       if( front ==-1)
            front = rear =0;
       else if (front >0)
            front --;
       else
       {     for (i = rear; i>=front; i--)
             {
                q[~~rear~~ i +1] = q[~~rear~~ i];
                 ~~rear++;~~
             }
       }     rear++;
       q[front] = el;
   }
}
```

```
void insertr()
{
    if(rear == max-1 && front == 0)
    {
        printf("full");
    }
    else
    {
        printf("enter element");
        scanf("%d", &el);
        ~~if(rear < max-1)~~
        ~~{~~
            ~~rear++;~~
            ~~q[rear] = el;~~
        }
        if(front == -1)
        {
            front = rear = 0;
            q[rear] = el;
        }
        else if(rear < max-1)
        {
            rear++;
            q[rear] = el;
        }
        else
        {
            for(i = front; i <= rear; i++)
            {
                q[~~front~~ i - 1] = q[~~front~~ i];
                ~~front++~~
            } q[rear] = el;
            }
            front--;
    } }*
}
```

```
void deletef( )
{  ~~int~~
  if ( front == -1)
  {
     printf(" empty");
  }
  else
  {
     if (front == rear)
     {
        printf(" deleted element is %d", q[front]);
        ~~for~~ ~~print~~ front = rear = -1;
     }
     ~~else if (front == max-1);~~
     {
        ~~front = 0;~~
     }  printf(" deleted element is %d", q[front]);
     else  front = 0;
     {
        printf(" deleted element is %d", q[front]);
     }  front = front+1;
     } ~~printf(" deleted element is %d", q[front]);~~
  }
}
void deleter ( )
{
  if (front == -1)
     printf(" empty");
  else
  {
     if (front == rear)
     {  printf(" deleted element is %d", q[rear]);
     }  front = rear = -1
     else if (rear == 0)
     {  printf(" deleted element is %d", q[rear]);
     }  rear = max-1;
     else
     {  printf(" deleted element is %d", q[rear]);
     }  rear = rear-1;
     ~~printf(" deleted element is %d", q[rear]);~~
  }
}
```

```
void display( )
{
    int i;
    for (i = front ; i <= rear ; i++)
        printf("%d", q[i]);
}
void main( )
{
  int ch;
  while ~~(1)~~ (1);
  {
   printf(" enter\n 1 to insert front\n
              2 to insert from rear\n
              3 to delete from front \n
              4 to delete from rear \n
              5 to display\n
              6 to exit");
   scanf("%d", &ch);
   switch (ch)
   {
       case 1: insertf(); break;
       case 2: insertr(); break;
       case 3: deletef(); break;
       case 4: deleter(); break;
       case 5: display(); break;
       case 6: exit(0);
   }
  }
}
```