CIRCULAR LINKED LIST

- Insert as a first node
- Insert as a last node
- Delete first node
- Delete last node
- Insert after a node
- Insert before a node
- Search
- Traverse

INSERT AS A FIRST NODE

```
void insertf()
 {
    struct studinfo  *newnode,*ptr;
    newnode=(struct studinfo *) malloc(sizeof(struct studinfo));
    printf("Enter a new  record : marks and name ");
    scanf("%d%s",&newnode->marks,newnode->name);

    if(start==NULL)
            {
                    start = newnode;
                    newnode->next = newnode;
            }
            else
              {
              ptr = start;
              while(ptr->next !=start)
                        ptr = ptr->next;
             newnode->next = start;
              start = newnode;
              ptr->next =start;
      }
}
```

INSERT AS A LAST NODE

```
void insertl()
{
    struct studinfo *newnode, *ptr;
    newnode=(struct studinfo *) malloc(sizeof(struct studinfo));
    printf("Enter a new  record : marks and name ");
    scanf("%d%s",&newnode->marks,newnode->name);
    if (start == NULL)
    {
            start =newnode;
            newnode->next = newnode;
    }
    else
    {
            ptr =start;
            while (ptr->next != start)
                    ptr= ptr->next;
            ptr->next = newnode;
            newnode->next = start;
    }
}
```

INSERT AFTER A NODE

```
void inserta()
 {
  int cnt=1, no;
    struct studinfo *ptr,*prevptr, *newnode;
     printf("\n enter number ...");
     scanf("%d",&no);
     ptr=start;
     while (cnt != no)
     {
     ptr = ptr->next;
     cnt++;
     }
    newnode=(struct studinfo *) malloc(sizeof(struct studinfo));
    printf("Enter a new  record : marks and name");
    scanf("%d%s",&newnode->marks,newnode->name);
    newnode->next = ptr->next;
    ptr->next = newnode;
}
```

INSERT BEFORE A NODE

```
void insertb()
 {

    int cnt=1, no;
    struct studinfo *ptr,*prevptr, *newnode;
     printf("\n enter number ...");
     scanf("%d",&no);
     ptr=start;
    if(no==1)
     {
            insertf();
     }
     else
     {
```

CONTINUE…

```
while(cnt !=no)
        {
                prevptr=ptr;
                ptr = ptr->next;
                cnt++;
        }
        newnode=(struct studinfo *) malloc(sizeof(struct
   studinfo));
        printf("Enter a new  record : marks and name " );
        scanf("%d%s",&newnode->marks,newnode->name);
        newnode->next=ptr;
        prevptr->next=newnode;
    }
}
```

DELETE FIRST NODE

```
void deletef()
 {
   struct studinfo *ptrs,*ptr;
   ptrs=start;
   ptr = start;
   if (start == NULL)
                printf("\n List is empty, element can not be delete");
     else
   {
   if (ptr->next == start)
                              start = NULL;
   else
   {
        while(ptr->next !=start)
                                ptr = ptr->next;
      start=start->next;
      ptr->next = start;
   }
   free(ptrs);
  }
}
```

DELETE LAST NODE

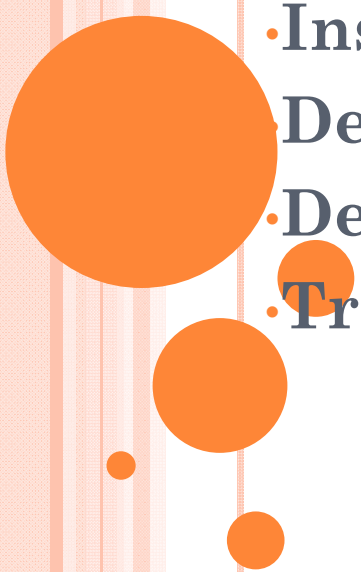
```
void deletel()
 {
      struct studinfo *ptr,*prevptr;
      ptr=start;
      if (start == NULL)
                  printf("\n List is empty, element can not be delete");
      else
      {
      if (ptr->next == start)
                   start = NULL;
      else
      {
                  while(ptr->next!=start)
                   {
                                prevptr=ptr;
                                ptr=ptr->next;
                   }
                   prevptr->next =start;
      }
       free(ptr);
  }
}
```

TRAVERSE

```
void traverse()
{
    struct studinfo *ptr;
    ptr= start;
    if (ptr)
    {
            while (ptr->next !=start)
            {
            printf("\nRecord: marks and name %d  %s\n",ptr->marks, ptr-
    >name);
                        ptr = ptr->next;
            }
            if (ptr)
               printf("\nRecord: marks and name %d  %s\n",ptr->marks, ptr-
    >name);
    }
    else
            printf("\nCircular list is empty ");
    getch();
}
```

Doubly Linked List

- Insert as a first node
- Insert as a last node
- Insert after a node
- Delete first node
- Delete last node
- Traverse



CRAETE

```
struct studinfo{
   int marks;
   struct studinfo *next;
   struct studinfo *prev;
   }*start;
```

INSERT AS A FIRST NODE

```
void insertf()
 {
   struct studinfo *newnode, *ptr;
   newnode=(struct studinfo *) malloc(sizeof(struct studinfo));
   printf("Entera new number : marks ");
   scanf("%d",&newnode->marks);
   newnode->next =NULL;
   newnode->prev=NULL;
   newnode->next=start;
   start->prev=newnode
   start=newnode;
}
```

INSERT AS A LAST NODE

```
void insertl()
{
    struct studinfo *newnode, *ptr;
    newnode=(struct studinfo *) malloc(sizeof(struct studinfo));
    printf("Entera new number : marks ");
    scanf("%d",&newnode->marks);
    newnode->next =NULL;
    newnode->prev=NULL;
    if (start == NULL)
            start =newnode;
    else
    {
    ptr = start;
    while (ptr->next!=NULL)
            ptr = ptr->next;
    ptr->next = newnode;
    newnode->prev=ptr;
    }
}
```

INSERT AFTER A NODE

```
void inserta()
 {
  int no,cnt=1;
  struct studinfo *newnode, *ptr;
  printf("insert a node number");
  scanf("%d",&no);
  ptr=start;
  while((no!=cnt) &&(ptr!=NULL))
   {
        ptr=ptr->next;
        cnt++;
   }
```

```
newnode=(struct studinfo *) malloc(sizeof(struct
   studinfo));
   printf("Enter a new number : marks ");
   scanf("%d",&newnode->marks);

   newnode->prev=ptr;
   newnode->next=ptr->next;
   ptr->next->prev= newnode;
   ptr->next = newnode;

}
```

DELETE FIRST NODE

```
void deletef()
 {
   struct studinfo *ptr;
   if(start==NULL)
   {
         printf("list is empty");
         return;
   }
   ptr=start;
   start=start->next;
   start->prev=NULL;
   free(ptr);
}
```

DELETE LAST NODE

```
void deletel()
 {
   struct studinfo *ptr,*prevptr;
   ptr=start;
   while(ptr->next!=NULL)
   {
         prevptr=ptr;
         ptr=ptr->next;
   }
   prevptr->next=NULL;
   free(ptr);

}
```

TRAVERSE

```
void traverse()
{
   struct studinfo *ptr;
   ptr= start;
   while (ptr!=NULL)
   {
   printf("\t %d",ptr->marks);
   ptr = ptr->next;
   }
   getch();
}
```