05/02/2021

Doubly Linked List;

```
struct node
{
int data;
struct node *next;
struct node *prev;
};

struct node *head=NULL, *newnode, *temp, *tail;

void create()
{
  int i, n;
  printf("Enter the no of elements");
  scanf("%d", &n);
for(i=0; i<n; i++)
  {
  newnode = (struct node*) malloc(sizeof);
   printf("Enter data:");
   scanf("%d", &newnode->data);
  newnode->prev = NULL;
  newnode->next = NULL;

  if(head==NULL)
    {
      temp=head=newnode;
    }
else {
  temp->next = newnode;
    newnode->prev = temp;
      temp = newnode;
}
}
```

```
void display ()
{
temp = head;
while(temp != NULL)
{
Pf("%d", temp->data);
temp = temp->next;
}
}

Void Insert_beg()
{
newnode = () malloc();
Pf("Enter new Element");
Sf("%d", & newnode->data);
   newnode->prev = NULL;
   head->prev = newnode;
   new->next = head;
       head = newnode;
  }
void
Insert_end()
{
newnode = () malloc();
Pf(Enter new Element);
Sf("%d", & newnode->data);
   tail->next = newnode;
   newnode->prev = tail;
   tail = newnode;
}
```

```
void insert_pos()
{
 int pos, i=1;
newnode = (*) malloc();
 Pf ("Enter the pos");
 Sf ("%d", &pos);
 if (pos<0)
 {
 Pf (invalid);
 }
 else {
 temp = head;
 while (i<pos-1)
 {
    temp = temp->next;
    i++;
 }
 Pf (Enter new element);
 Sf ("%d", &newnode->data);
 newnode->prev = temp;
 newnode->next = temp->next;
    temp->next = newnode;
 newnode->next->prev = newnode;
 }
}

Void del_beg()
{
  temp = head;
 if (head == NULL)
 {
 Pf (List is empty);
 }
```

```
else {
    head = temp -> next;
    head -> prev = NULL;
    free(temp);
}
}

void del_end()
{
    temp = tail;
    if (tail == NULL)
    {
        pf("List is empty");
    }
else {
    tail -> prev -> next = NULL;
    tail = tail -> prev;
    free(temp);
}
}

void del_pos()
{
    int pos, i = 1;
pf("Enter the pos");
sf("%d", &pos);
temp = head;
while (i < pos-1)
{
    temp = temp -> next;
    i++;
}
temp -> prev -> next = temp -> next;
```

```
void main()
{
int choice;
while(1){
  Pf("Enter operation: \n
        1. create\n
        2. display\n
        3. insert_beg\n
        4. insert_end\n
        5. insert_pos\n
        6. delete_beg\n
        7. delete_end\n
        8. delete_pos\n
        9. -1 to end\n");
if("%d", &choice);
  if (choice == -1)
  {
    Pf("operation completed");
  }
else{
switch(choice) {
Case 1: create();
        break;
Case 2:
Case 3
Case 4
Case 5
Case 6
Case 7
Case 8:
default: Pf("Invalid choice");
}
}
}
```

O/P;

Enter operation
1. Create
2. display
3. Insert at left
4. delete at position
5. -1 to End
nter Operation

ter the no of elements:
3
ter the 1 element:
1
ter the 2 element:
2
ter the 3 element:
3
ter operation

ter operation
4
ter the node
3
ter data
1
er operation
2

enter operation
4
enter position
3
enter operation
2
1
2
3

```
enter operation
1.create
2.display
3.insert at left
4.delete at position
5.-1 to end
enter operation
1
Enter the no. of elements:
3
Enter the 1 element :
2
Enter the 2 element :
3
Enter the 3 element :
4
enter operation
2
2
3
4
enter operation
3
enter the node
1
enter data
5
enter operation
2
5
```

```
2
3
4
enter operation
3
enter the node
3
enter data
6
enter operation
2
5
2
6
3
4
enter operation
4
enter position
3
enter operation
2
5
2
3
4
enter operation
-1
completed
```