R.O
Date 22 01 2024

1. WAP to implement singly linked list with following operations.

a) Create a linked list.
b) Insertion of a node at first position, at any position and at end of list. Display the contents of the linked list.

```
#include <stdio.h>
#include <stdlib.h>
Struct node
{
  int data;
  Struct node *next;
};
Struct node *start = NULL;
struct node *create_ll (struct node *);
struct node *display (struct node *);
struct node *insert_beg (struct node *);
struct node *insert_end (struct node *);
struct node *insert_after (struct node *);
struct node *sort_list (struct node *);
```

```
int main()
{
 int choice;
 printf("\n *** menu ***\n 1.create a digit \it
2. display \it 3. insert_beg \it 4. insert_end \it
5. insert_before \it 6. insert_after");
 do
 {
   printf("\n Enter the choice: ");
   scanf("%d", &choice);
   switch (choice)
   {
     case 1: start = create_ll(start);
             printf("\n linked digit created");
             break;
     case 2: start = display(start);
             break;
     case 3: start = insert_beg(start);
             break;
     case 4: start = insert_end(start);
             break;
     case 5: start = insert_before(start);
             break;
     case 6: start = insert_after(start);
             break;
```

```
while (choice != 7);
return 0;
}

struct node *create_dll (struct node *start)
{
 struct node *new_node, *ptr;
 int num;
 printf ("\n enter -1 to end");
 printf ("\n enter the data: ");
 scanf ("%d", &num);
 while (num != -1)
 {
  new_node = (struct node *) malloc (sizeof (struct node));
  new_node -> data = num;
  if (start == NULL)
  {
   new_node -> next = NULL;
   start = new_node;
  }
  else
  {
   ptr = start;
   while (ptr -> next != NULL)
    ptr = ptr -> next;
```

```
        ptor -> next = new_node;
        new_node -> next = NULL;
    }
    printf("\n Enter the data: ");
    scanf("%d", &num);
  }
return start;
};

struct node *display(struct node *start)
{
 struct node *ptor;
 ptor = start;
 while (ptor != NULL)
 {
   printf("\t %d", ptor -> data);
   ptor = ptor -> next;
 }
return start;
};
```

```
struct node *ingcrt_beg (struct node *start)
{
 struct node *new_node;
 int num;
 printf ("\n Enter the data: ");
 scanf ("%d", &num);
 new_node = (struct node *) malloc (sizeof (struct node));
 new_node -> data = num;
 new_node -> next = start;
 start = new_node;
 return start;
};

Struct node *ingcrt_end (struct node *start)
{
 struct node *new_node, *ptr;
 int num;
 printf ("\n Enter the data: ");
 scanf ("%d", &num);
 new_node = (struct node*) malloc (sizeof (struct node));
 new_node -> data = num;
 new_node -> next = NULL;
 ptr = start;
 if ( start != NULL)
 {
  start = new_node;
}
```

```
else
{
while (ptor -> next != NULL)
        ptor = ptor -> next;
    ptor -> next = new_node;
return start;
}
};

struct node * Ingcert_before (struct node *start)
{
    struct node *new_node, *ptor, *preptor;
    int num, val;
    printf ("\n Enter the data: ");
    scanf ("%d", &num);
    printf ("\n Enter the value before which data ha
do be Ingerted: ");
    scanf ("%d", &val);
    new_node -> data = num;
    ptor = start;
    while (ptor -> data != val)
    {
     preptor = ptor;
     ptor = ptor -> next;
    }
```

```
preptor -> next = new_node;
new_node -> next = ptor;
return start;
};

struct node * Insert_after (struct node * start)
{
  struct node *new_node, *ptor, *preptor;
  int num, val;
  printf ("\n enter the data:");
  scanf ("%d", &num);
  printf ("\n enter the value after which data has to
be Inserted:");
  scanf ("%d", &val);
  new_node = (struct node*) malloc (size of (struct node));
  new_node -> data = num;
  ptor = start;
  preptor = ptor;
  while (preptor -> data != val)
  {
    preptor = ptor;
    ptor = ptor -> next;
  }.
  preptor -> next = new_node;
  new_node -> next = ptor;
  return start;
};
```

```
struct node * delete_beg (struct node * start)
{
  struct node *ptr;
  ptr = start;
  start = start -> next;
  free (ptr);
  return start;
};

struct node *delete_end (struct node *start)
{
  struct node *ptr, *preptr;
  ptr = start;
  while (ptr -> next != NULL)
  {
    preptr = ptr;
    ptr = ptr -> next;
  }
  preptr -> next = NULL;
  free (ptr);
  return start;
};
```

```
struct node *delete_node (struct node *start)
{
  struct node *ptr, *preptr;
  int val;
  printf ("In Enter the value to be deleted: ");
  scanf ("%d", &val);
  ptr = start;
  if (ptr -> data == val)
  {
    start = delete_beg (start);
    return start;
  }.
  else {
    while (ptr -> data != val)
    {
      preptr = ptr;
      ptr = ptr -> next;
    }.
  preptr -> next = ptr -> next;
  free (ptr);
  return start;
  }
};
```

```
struct node *delete_after(struct node * start);
{
 struct node *ptr, *preptr;
 int val;
 printf("\n Enter the value after which the node has
to be deleted: ");
 scanf("%d", &val);
 ptr = start;
 preptr = ptr;
 while (preptr -> data != val)
 {
    preptr = ptr;
    ptr = ptr -> next;
 }
 preptr -> next = ptr -> next;
 free(ptr);
 return start;
};
```

```
struct node *delete_list (struct node *start)
{
  struct node *ptr;
  if (start != NULL)
  {
    ptr = start;
    while (ptr != NULL)
    {
      printf ("%d is deleted\n", ptr->data);
      start = delete_beg (ptr);
      ptr = start;
    }
  }
  return start;
};
```

~~struct node *sort~~

Output:

--menu---

1. Create linked list
2. display
3. insert - beg
4. ~~Insert - beg~~
5. insert - end
6. Insert - before
7. Insert - after
8. del - beg
9. del - end
10. del - node
11. exit

Enter your choice : 1

Enter -1 to exit.

Enter the num : 2
Enter the num : 3
Enter the num : 4
Enter the num = 5
Enter the num = -1

linked list created enter the choice 9.

:2 3 4 5 6.

Enter the choice: 10

exit.

deletion

o/p

Sun 22/1/24

Enter the choice: 7

Enter the choice: 2

2 3 4

Enter the choice: 7

Enter the choice: 2

2 3

Enter the choice: 9.

enter the value that has to be deleted: 2.

Enter the choice: 2

3.