Week-5 Single Linked List (creating LL, Inserting an element)

```
#include <stdio.h>
#include <malloc.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_before (struct node*);
struct node *insert_after (struct node*);
struct node *sort_list (struct node*);
struct node *sort_list (struct node *);
int main(){
 int choice;
 printf ("\n 1. create a list. \n 2. display \n 3. Insert_beg
      \n. 4. Insert_end \n 5. Insert_before. \n 6. Insert_after");
 do{
 pf("\n Enter the choice:");
 sf("%d", &choice);
 switch (choice){
   case 1: start = create_ll(start);
           printf("\n LL 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 = sort_list (start);
           break;
   }
 }while (choice != 6);
 return 0;
}
struct node * create_ll (struct node *start){
  struct node *new_node, *ptr;
 int num;
 printf(" Enter -1 to end");
 pf("\n Enter the data");
 sf("%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;
      ptr -> 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 *ptr;
  ptr = start;
  while (ptr != NULL){
    pf(" %d", ptr->data);
    ptr = ptr->next;
  }
  return start;
};
struct node * insert_beg (struct node *start)
{
  struct node * new_node;
  int num;
  pf(" enter the data");
  sf("%d", &num);
  new_node = (struct node *) malloc (size of (struct node));
  new_node -> data = num;
  new_node -> next = start;
  start = new_node;
  return start;
};
```

```
struct node * insert_end (struct node * start)
{
  struct node * new_node, *ptr;
  int num;
  printf ("\n enter the data:");
  scanf ("%d", &num);
  new_node -> data = num;
  new_node -> next = NULL;
  ptr = start;
  if (start == NULL){
    start = new_node;
  } else{
    while (ptr->next != NULL){
      ptr = ptr -> next;
    }
    ptr -> next = new_node;
    return start;
  }
};
struct node * insert_before (struct node* start){
  struct node * new_node, *ptr, * preptr;
  int num, val;
  pf(" \n Enter the data");
  sf("%d", &num);
  pf("\n enter the data before which value to be
      inserted");
  sf("%d", &val);
  new_node = (struct node*) malloc (sizeof (struct node));
  new_node -> data = num;
  ptr = start;
```

```
while (ptr -> data != val){
    preptr = ptr;
    ptr = ptr -> next;
}
preptr -> next = new_node;
new_node -> next = ptr;
return start;
};
struct node *insert_after (struct node *start){
    struct node *new_node, *ptr, *preptr;
    int val, num;
    printf("\n enter the data");
    scanf("%d", &num);
    printf("\n enter the value after which data to be inser
    -ted");
    scanf("%d", &val);
    new_node = (struct node *) malloc (sizeof (struct node));
    new_node -> data = num;
    ptr = start;
    preptr = ptr;
    while (preptr -> data != val)
    {
        preptr = ptr;
        ptr = ptr -> next;
    }
    preptr -> next = new_node;
    new_node -> new = ptr;
    return start;
};
```

```
struct node * sort_list (struct node * start){
    struct node *ptr1, *ptr2;
    int temp;
    ptr1 = start;
    while (ptr1 -> next != NULL){
        ptr2 = ptr1 -> next;
        while (ptr2 != NULL){
            if (ptr1 -> data > ptr2 -> data)
            {
                temp = ptr1 -> data;
                ptr1 -> data = ptr2 -> data;
                ptr2 -> data = temp;
            }
            ptr2 = ptr2 -> next;
        }
        ptr1 = ptr1 -> next;
    }
    display (start);
};
```

2) WAP to (i) create a Linked List
(ii) Deletion of (a) First element
(b) Last element
(c) specified element.

```
#include <stdio.h>
#include <malloc.h>
struct node
{
 int data;
 struct node *next;
}
struct node *start = NULL;
struct node *createll (struct node*);
struct node *display (struct node*);
struct node *delete_beg (struct node*);
struct node *delete_end (struct node*);
struct node *delete before (struct node*);
struct node *delete_after (struct node*);
struct node *sort_list (struct node *);
int main(){
 int choice;
 1. create List 6. display 7. sort list 8. exit
 pf(" 1. del_beg . 2. del_end . 3. del_before 4. del_after");
 sf("%d", &choice);
int choice;
pf(" 4 create_ll 5. display 1. delete_beg
 2. delete_end 3. delete_node");
sf("%d", &choice);
```

```
switch (choice){
case 1: delete_beg (start);
        break;
case 2: delete_end (start);
        break;
case 3: delete node before (start);
        break;
case 4: delete_after (start);
        break;
case 4: create_ll (start);
        pf(" Linked List created");
        break;
case 5: display (start);
        break;
case 7: delete_list (start);
        break;
case 8: exit (0
}
}while (choice != 8);
return 0;
}
```

```
struct node * create_ll (struct node *start){
  struct node *new_node, *ptr;
  int num;
  pf("enter -1 to end");
  pf("enter the data:");
  sf("%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;
  ptr -> next = new_node;
  new_node -> next = NULL;
}
return start;
};
```

```
struct node * display (struct node *start){
  struct node *ptr;
  ptr = start;
  while (ptr != NULL){
    pf("\t %d", ptr -> data);
    ptr = ptr -> next;
  }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;
 pf(" enter the value to be deleted");
 sf ("%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;
}
```

22/01/2024 Insertion programme

o/p :- 1. create a list 2. display 3. Insert_beg
4. Insert_end 5. Insert_bef 6. Insert_after

enter choice : 1

enter -1 to end

enter the data: 1 2 3 4

enter the data : -1

List created

enter the choice: 3

Enter the data : 0

Enter the choice: 4

enter the data : 5

enter the choice : 2

0 1 2 3 4 5.

Enter the choice 7.

returned 0.

Deletion program

o/p :

1. create a list 2. display 3. delete-beg 4. delete-end
5. delete-node

Enter the choice: 1

enter -1 to end

enter the ~~choice~~ data: 1 2 3 4 5 6 7

enter the data -1

enter the choice 2

1 2 3 4 5 6 7

enter the choice 3

enter the choice 4

enter the choice 5

enter the value to be deleted. 3

enter the choice 2

2 4 5 6.

22/01/24