Linked list

→

```
#include <stdio.h>
#include <stdlib.h>

void createC);
void display C);
void insert_begin C);
void insert_end C);
void insert_pas C);
void delete_pos C);
void delete_all C);
void count_elements C);
void reverse_list C);

Struct node {
  int info;
  Struct node * next;
};
struct node * start = NULL;

int main C) {
int choice;
while C1) {
 printf(" In 1. Create In");
printf(" 2. Display In");
printf(" 3. insert at begin In");
printf(" 4. insertt at end In");
printf(" 5. insert at any position In");
printf(" 6. delete at position In");
 printf(" 7. delete all In");
```

```
printf(" 8. count elements \n");
printf(" 9. reverse list \n");
printf("10. exit \n");

printf("Enter your choice :");
scanf("%d", &choice);

switch(choice){
  case 1: create();
          break;
  case 2:
          display();
          break;
  case 3:
          insert_begin();
          break;
  case 4:
          insert_end();
          break;
  case 5:
          insert_pos();
          break;
  case 6:
          delete_pos();
          break;
  case 7:
          delete_all();
          break;
  case 8:
          count_elements();
          break;
  case 9:
          reverse_list();
```

```
            break;
        case 10:
            exit(0);
        default:
            printf("Incorrect choice. Choose from
                    the given numbers.\n");
    }
}

    return 0;
}

void create(){
    struct node *temp, *ptr;
    temp = (struct node*) malloc(sizeof(struct node));
    printf("Enter data:");
    scanf("%d", &temp->info);
    temp->next = NULL;
    if(start == NULL){
        start = temp;
    }else{
        ptr = start;
        while(ptr->next != NULL){
            ptr = ptr->next;
        }
        ptr->next = temp;
    }
}
```

```
void display(){
    struct node *ptr = start;
    if(start == NULL){
    printf("\n Empty list \n");
    return;
}
  printf("\n List elements :");
  while (ptr != NULL){
  printf(" %d ", ptr -> info);}
    ptr = ptr -> next;
}
  printf("\n");
}

void insert_begin(){
  struct node *temp;
  temp = (struct node*) malloc (sizeof(
    struct node));
  printf("Enter data : ");
  scanf("%d", &temp -> info);
  temp -> next = start;
  start = temp;
}

void insert_end(){
   create();
}

void insert_pos(){
struct node *temp, *ptr;
int pos, i;
temp = (struct node*) malloc (size of (struct node));
```

```
        printf("Enter data:");
        scanf("%d", &temp->info);
        temp->next = NULL;

    printf("Enter position to insert:");
    scanf("%d", &pos);

    if(pos == 1){
        temp->next = start;
        start = temp;
        return;
    }
    ptr = start;
    for(i = 0; i < pos-2; i++){
        if(ptr == NULL){
            printf("Position not found.\n");
            free(temp);
            return;
        }
        ptr = ptr->next;
    }
    if(ptr == NULL){
        printf("Position not found\n");
        free(temp);
        return;
    }

    if(temp == Null){
        printf("Position not found.\n");
        return;
    }
    ptr->next = temp->next;
    free(temp);
```

```
}
void delete_all(){
struct node *ptr;
while (start != NULL){
    ptr = start;
    start = start -> next;
    free(ptr);
}
printf("\n All nodes deleted. List is now emp
}

void count_elements(){
struct node *ptr = start;
int count = 0
while (ptr != NULL){
count ++;
ptr = ptr -> next
}
printf(" No. of elements in list: %d \n", count)
}

void reverse_list(){
    struct node * prev = NULL * current = start,
    * next = NULL;

while (current != NULL){
next = current -> next;
prev = current;
current = next;
}
start = prev;
printf("List reversed successfully. \n");
}
```

O/P:

1) Create
2) Display
3) insert at begin
4) insert at end
5) insert at any position
6) Delete all
7) delete at position
8) Count elements
9) Reverse list
10) Exit

Enter your choice : 1
Enter your data : 2

Menu...
Enter your choice : 3
Enter your data : 4

menu...
Enter your choice : 5
Enter your data : 8
Enter position to insert : 2

Menu...
Enter your choice : 6
Enter position to delet: 2

menu...
Enter your choice : 8
No. of elements in list : 3

Menu:--
Enter your choice: 2
List elements: 6 2 4

Menu....
Enter your choice: 7
All nodes delete. list is now empty.

Menu...
Enter your choice: 2
Empty list

Menu....
Enter your choice: 10

* Theory questions:-

Q.1) Explain basic terminologies of singly linked list with examples

→ Basic terminologies -

1) Node
→ Basic building block of linked list
→ Constains two parts.
→ Data → Actual parts.
→ Next → address of next node.

2) Head
→ A special pointer that always stores the address of first node.

s) NULL Pointer
→ The next field of the last node contains NULL to mark the end.

u) Address (Link)
→ Each node is stored at a different memory location and the pointer stores the address of next node

s) Empty list.
→ It is a linked list with no nodes.

Q2) Diff two array & linked list.

Feature	Array	Linked list
Memory allocation	Contiguous	Non-contiguous
Size	Fixed at run time	Dynamic, can grow
Accessing elements	Direct access with index	Sequential traversal
Insertion/ Deletion	Requires shifting elements	Easy, just change links.
Extra Memory	No overhead	Extra pointer per node

13/11