

- i) 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 content of linked list.
 - c) Deletion of first element, specified element and last element in the list.

```

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

```

```
    printf("\nEnter the data:");
    scanf("%d", &num);
}
return start;
};
```

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

```
struct node *insert_beg (struct node *start)
{
    struct node *new_node;
    int num;
    printf("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 *insert_end (struct node *start)
{
    struct node *new_node, *ptr;
    int num;
    printf("Enter the data: ");
    scanf("%d", &num);
    new_node = (struct node *) malloc(sizeof(struct node));
    if (start == NULL)
```

```

new-node->next = NULL;
start = new-node;
}
else
{
    *ptr = start;
    while (ptr->next != NULL)
    {
        ptr = ptr->next;
    }
    ptr->next = new-node;
}
}

```

struct node *insert_before (struct node *start)

```

{
    struct node *new-node, *ptr, *preptr;
    int num, val;
    printf ("enter the data:");
    scanf ("%d", &num);
    printf ("enter the value before which data has to be inserted:");
    scanf ("%d", &val);
    new-node = (struct node*)malloc(sizeof(struct node));
    new-node->data = num;
    preptr = 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)

```

{
    while (preptr->data != val)
    {
        preptr = ptr;
        ptr = ptr->next;
    }
}
```

```
    pptr->next = new_node;
    new_node->next = pptr;
    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, * pptr;  
    int val;  
    printf ("Enter the value to be deleted: ");  
    scanf ("%d", &val);  
    pptr = start;  
    if (ptr->data == val)  
    {  
        start = delete_beg (start);  
    }  
    return start;  
};
```

```

else{
    while (ptr->data != start);
    {
        preptr = ptr;
        ptr = ptr->next;
    }
    preptr->next = ptr->next;
    free(ptr);
    return start;
}
}

```

op:

~~main~~

1. create linked list

2. display

3. insert-beg

4. insert-end

5. insert-before

6. insert-after

7. del-beg

8. ~~del~~-end

9. del-node

10. exit

enter the choice

(

Enter -1 to exit

Enter the number : 40

Enter the number : 20

Enter the number : -1

Linked list created -

enter the choice:

2

40 20

~~enter choice~~

enter choice : 3

10

enter choice : 2

10 40 20

enter choice : 4

~~30~~

enter choice : 2

10 40 20 30

enter choice : 7

deleted element : ~~10~~

enter choice : 2

40 20 30

enter choice : 8

deleted element : 30

enter choice : 2

40 20

enter choice : 9

- enter element : 40

enter choice : 2

20

white
O/P
completely

(B)
22/12/24

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 struct node
5 {
6     int data;
7     struct node *next;
8 };
9
10 struct node *head = NULL;
11
12 struct node *create_ll(struct node *head)
13 {
14     struct node *new_node, *ptr;
15     int num;
16     printf("Enter -1 to exit.. \n");
17     printf("Enter the num: ");
18     scanf("%d", &num);           I
19
20     while (num != -1)
21     {
22         new_node = (struct node *)malloc(sizeof(struct node));
23         new_node->data = num;
24         if (head == NULL)
25         {
26             head = new_node;
27             new_node->next = NULL;
28         }
29         else
30         {
31             ptr = head;
32             while (ptr->next != NULL)
33             {
34                 ptr = ptr->next;
35             }
36             ptr->next = new_node;
37             new_node->next = NULL;
38     }
```

```
id_semi > C learning > C linkedlist2.c > create_ll(node *)  
    new_node->next = NULL,  
38  
39     }  
40     printf("Enter the num: ");  
41     scanf("%d", &num);  
42     return head;  
43 }  
44  
45 struct node *insert_beg(struct node *head)  
46 {  
47     struct node *new_node;  
48     int num;  
49     printf("Enter the num: ");  
50     scanf("%d", &num);  
51     new_node = (struct node *)malloc(sizeof(struct node));  
52     new_node->data = num;  
53     if (head == NULL)  
54     {  
55         head = new_node;  
56         new_node->next = NULL;  
57     }  
58     else  
59     {  
60         new_node->next = head;  
61         head = new_node;  
62     }  
63     return head;  
64 }  
65  
66 struct node *insert_end(struct node *head)  
67 {  
68     struct node *ptr, *new_node;  
69     int num;  
70     printf("Enter the num: ");  
71     scanf("%d", &num);  
72     new_node = (struct node *)malloc(sizeof(struct node));  
73     new_node->data = num;  
74     new_node->next = NULL;
```

```
74     new_node->next = NULL;
75     if (head == NULL)
76     {
77         head = new_node;
78     }
79     else
80     {
81         ptr = head;
82         while (ptr->next != NULL)
83         {
84             ptr = ptr->next;
85         }
86         ptr->next = new_node;
87     }
88     return head;
89 }
90
91 struct node *insert_before(struct node *head)
92 {
93     struct node *new_node, *ptr, *prevptr;
94     int num, val;
95     printf("Enter the num: ");
96     scanf("%d", &num);
97     printf("Enter the value before which number has to be inserted: ");
98     scanf("%d", &val);
99     new_node = (struct node *)malloc(sizeof(struct node));
100    new_node->data = num;
101    ptr = head;
102    while (ptr->next != NULL)
103    {
104        prevptr = ptr;
105        ptr = ptr->next;
106    }
107    prevptr->next = new_node;
108    new_node->next = ptr;
109    return head;
110 }
```

```
110 }
111 struct node *insert_after(struct node *head)
112 {
113     struct node *new_node, *ptr, *prevptr;
114     int num, val;
115     printf("Enter the num: ");
116     scanf("%d", &num);
117     printf("Enter the value before which number has to be inserted: ");
118     scanf("%d", &val);
119     new_node = (struct node *)malloc(sizeof(struct node));
120     new_node->data = num;
121     ptr = head;
122     while (ptr->next != NULL)           I
123     {
124         prevptr = ptr;
125         ptr = ptr->next;
126     }
127     prevptr = ptr;
128     ptr = ptr->next;
129     prevptr->next = new_node;
130     new_node->next = ptr;
131     return head;
132 }
133 struct node *display(struct node *head)
134 {
135     struct node *ptr;
136     if (head == NULL)
137     {
138         printf("Linked List is empty...\n");
139     }
140     else
141     {
142         ptr = head;
143         while (ptr != NULL)
144         {
145             printf("%d ", ptr->data);
146             ptr = ptr->next;
147         }
148     }
149 }
```

```
147     }
148     printf("\n");
149 }
150 return head;
151 }
152
153 struct node *delete_beg(struct node *head)
154 {
155     struct node *ptr;
156     if (head == NULL)
157     {
158         printf("Nothing to delete.. \n");
159     }
160     else
161     {
162         ptr = head;
163         head = ptr->next;
164         free(ptr);
165     }
166     return head;
167 }
168
169 struct node *delete_end(struct node *head)
170 {
171     struct node *ptr, *prevptr;
172     ptr = head;
173     while (ptr->next != NULL)
174     {
175         prevptr = ptr;
176         ptr = ptr->next;
177     }
178     prevptr->next = NULL;
179     free(ptr);
180     return head;
181 }
```

```
src_semi > C:\learning > C:\Users\DELL\PycharmProjects\linkedlist> python linkedlist.py
183     struct node *delete_node(struct node *head)
184 {
185     struct node *ptr, *prevptr;
186     int val;
187     printf("Enter the value that has to be deleted: ");
188     scanf("%d", &val);
189     ptr = head;
190     if (ptr->data == val)
191     {
192         head = delete_beg(head);      I
193         return head;
194     }
195     else
196     {
197         while (ptr->data != val)
198         {
199             prevptr = ptr;
200             ptr = ptr->next;
201         }
202         prevptr->next = ptr->next;
203         free(ptr);
204         return head;
205     }
206 }
207
208 int main()
209 {
210     int choice;
211     printf("-----menu-----\n");
212     printf("1.create lined list\n 2.display\n 3.insert_beg\n 4.insert_end\n 5.insert_before\n 6.insert_after\n 7.del_beg\n 8.del_end\n");
213     do
214     {
215         printf("enter the choice:\n");
216         scanf("%d", &choice);
217         switch (choice)
218         {
219             case 1:
```

```
213     do
214     {
215         printf("enter the choice:\n");
216         scanf("%d", &choice);
217         switch (choice)
218         {
219             case 1:
220                 head = create_ll(head);
221                 printf("linked list created");
222                 break;
223             case 2:
224                 head = display(head); I
225                 break;
226             case 3:
227                 head = insert_beg(head);
228                 break;
229             case 4:
230                 head = insert_end(head);
231                 break;
232             case 5:
233                 head = insert_before(head);
234                 break;
235             case 6:
236                 head = insert_after(head);
237                 break;
238             case 7:
239                 head = delete_beg(head);
240                 break;
241             case 8:
242                 head = delete_end(head);
243                 break;
244             case 9:
245                 head = delete_node(head);
246                 break;
247         }
248     } while (choice != 10);
249 }
```

-----menu-----

- 1.create lined list
- 2.display
- 3.insert_beg
- 4.insert_end
- 5.insert_before
- 6.insert_after
- 7.del_beg
- 8.del_end
- 9.del_node
- 10.exitenter the choice:

1

Enter -1 to exit..

Enter the num: 40

Enter the num: 20

Enter the num: -1

linked list createdenter the choice:

2

40 20

enter the choice:

3

Enter the num: 10

enter the choice:

2

10 40 20

enter the choice:

4

Enter the num: 30

enter the choice:

2

10 40 20 30

enter the choice:

7

enter the choice:

2

40 20 30

enter the choice:

8

enter the choice:

2

40 20

Enter the num: 20

Enter the num: -1

linked list createdenter the choice:

2

40 20

enter the choice:

3

Enter the num: 10

enter the choice:

2

10 40 20

enter the choice:

4

Enter the num: 30

enter the choice:

2

10 40 20 30

enter the choice:

7

enter the choice:

2

40 20 30

enter the choice:

8

enter the choice:

2

40 20

enter the choice:

9

Enter the value that has to be deleted: 40

enter the choice:

2

20

enter the choice:

10

Process returned 0 (0x0) execution time : 100.379 s
Press any key to continue.