1BM19CS216

Linked list Implementation.

a) Create a linked list

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

c) Deletion of first element, specified element and last element in the list.

d) Display the contents:

algorithm.

```
insert_front
temp = getnode();
temp->info = item;
temp->link = NULL;
if (first == NULL)
return temp;
temp->link = first;
first = temp
return first;
```

```
delete_front
if first == NULL
PRINT can't delete
temp = first;
temp = temp->link;
free(first);
```

```
insert rear.
temp = getnode();
temp->info = item;
temp->link = NULL;
if (first == NULL)
return temp
cur = first
while cur->link != NULL
cur = cur->link;
cur->link = temp;
```

```
delete rear.
if (first->link == NULL)
prev = cur;
cur = cur->link;
free cur
prev->link = NULL;
```

program:

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
int info;
struct node *link;
};
typedef struct node *NODE;
NODE getnode()
{
NODE x;
x = (NODE)malloc(sizeof(struct node));
if (x==NULL)
{
printf("memory full\n");
exit(0);
}
return x;
}

void freenode(NODE x)
{
free(x);
}

NODE insert front(NODE first,int item)
{
NODE temp;
temp = getnode();
```

```
temp->info = item;
temp->link = NULL;
if (first == NULL)
return temp;
temp->link = first;
first = temp;
return first;
}

NODE delete_front(NODE first)
{
NODE temp;
if (first == NULL)
{
printf("Item list is empty can't delete\n");
return first;
}
temp = first;
temp = temp->link;
printf("Item deleted at front-end is %d\n",
                first->info);
free(first);
return temp;
}

NODE insert_rear(NODE first, int item)
{
NODE temp, cur;
temp = getnode();
```

```
temp->info=item;
temp->link =NULL;
if (first ==NULL)
return temp;
cur=first;
while (cur->link!=NULL)
cur=cur->link;
cur->link =temp;
return first;
}

NODE delete_rear (NODE first)
{
NODE cur, prev;
if (first ==NULL)
{
printf ("List is empty can't delete \n");
return first;
}
if (first->link ==NULL)
{
printf ("Item deleted is %d \n", first->info);
free (first);
return NULL;
}
prev =NULL;
cur =first;
while (cur->link != NULL)
{
```

```
prev = cur;
cur = cur->link;
}
printf("item deleted at the rear-end is %d",
    cur->info);
free(cur);
prev->link = NULL;
return first;
}

NODE order-list (int item, NODE first)
{
NODE temp, prev, cur;
temp = getnode();
temp->info = item;
temp->link = NULL;
if (first == NULL)
  return temp;
if (item < first->info)
{
temp->link = first;
return temp;
}
prev = NULL;
cur = first;
while (cur != NULL && item > cur->info)
{
   prev = cur;
   cur = cur->link;
}
```

```
pre->link = temp;
temp->link = cur;
return first;
}

NODE delete_info(int key, NODE first)
{
NODE prev, cur;
if (first == NULL)
{
printf("The list is empty\n");
return NULL;
}

if (key == first->info)
{
cur = first;
first = first->link;
freenode(cur);
return first;
}

prev = NULL;
cur = first;
while (cur != NULL)
{
if (key == cur->info) break;
prev = cur;
cur = cur->link;
}
```

```
if (cur == NULL)
{
printf("Search is unsuccessfull \n");
return first;
}
prev->link = cur->link;
printf("The item deleted is %d", cur->info);
freenode(cur);
return first;
}

void display (NODE first)
{
NODE temp;
if (first == NULL)
printf("The list is empty can not display
        items. \n");
for (temp = first; temp != NULL; temp =
                temp->link)
{
printf ("%d \n", temp->info);
}
}
int main()
{
int item, choice, key;
NODE first = NULL;
for (;;)
{
```

```
{
printf("\n 1: Insert a node at the front
        \n 2: Delete from the front
        \n 3: Insert a node at the end
        \n 4: Delete from the end
        \n 5: Insert in a Orderly list
        \n 6: Delete item
        \n 7: Display the list
        \n 8: Exit");
printf("Enter your choice : ");
scanf("%d", &choice);
switch(choice)
{
case 1: printf("Enter the data to be inserted
          at the front : \n");
scanf("%d",&item);
first=insert_front(first,item);
break;
case 2 : first=delete_front(first);
break;
case 3: printf("Enter the data to be inserte
          at the end : \n");
scanf("%d",&item);
first=insert_rear(first,item);
break;
case 4: first=delete_rear(first);
break;
```

```
case 5 : printf(" Enter the data to be
         inserted in the list : \n" )
scan("%d", &item);
first = order_list (item, first);
break;
case 6 : printf ("Enter the data to be
         deleted :\n");
Scanf ("%d ", &key);
first = delete_info ( key, first );
break;
case 7 : display (list);
breeak;
default : exit(0);
break;
}
}
}
```