Linked_List.c

Saved

```c
#include<stdio.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("list is empty cannot 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. Cannot 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 rear-end is %d",cur->info);
    free(cur);
    prev->link=NULL;
    return first;
}
void display(NODE first)
{
    NODE temp;
    if(first==NULL)
        printf("List empty. Cannot display items!\n");
    for(temp=first;temp!=NULL;temp=temp->link)
    {
        printf("%d\n",temp->info);
    }
}
void main()
{
    int item,choice,pos;
    NODE first=NULL;
    for(;;)
    {
        printf("\n1.Insert_front\n2.Delete_front\n3.Insert_rear\n4.Delete_rear\n5.display_list\n6.Exit\n");
        printf("Enter Choice:");
        scanf("%d",&choice);
        switch(choice)
        {
        case 1:
            printf("Enter the item at front-end: ");
          scanf("%d",&item);
          first=insert_front(first,item);
          break;
        case 2:
            first=delete_front(first);
          break;
        case 3:
            printf("Enter the item at rear-end: ");
          scanf("%d",&item);
          first=insert_rear(first,item);
          break;
        case 4:
            first=delete_rear(first);
          break;
        case 5:
            display(first);
          break;
        case 6:
            exit(0);
          break;
        default:
            printf("Enter a Valid Choice!!");
        }
    }
}
```

```
Terminal

1.Insert_front
2.Delete_front
3.Insert_rear
4.Delete_rear
5.display_list
6.Exit
Enter Choice:1
Enter the item at front-end: 10

1.Insert_front
2.Delete_front
3.Insert_rear
4.Delete_rear
5.display_list
6.Exit
Enter Choice:1
Enter the item at front-end: 9

1.Insert_front
2.Delete_front
3.Insert_rear
4.Delete_rear
5.display_list
6.Exit
Enter Choice:3
Enter the item at rear-end: 20

1.Insert_front
2.Delete_front
3.Insert_rear
4.Delete_rear
5.display_list
6.Exit
Enter Choice:3
Enter the item at rear-end: 21

1.Insert_front
2.Delete_front
3.Insert_rear
4.Delete_rear
5.display_list
6.Exit
Enter Choice:5
9
10
20
21

1.Insert_front
2.Delete_front
3.Insert_rear
4.Delete_rear
5.display_list
6.Exit
Enter Choice:2
item deleted at front-end is=9

1.Insert_front
2.Delete_front
3.Insert_rear
4.Delete_rear
5.display_list
6.Exit
Enter Choice:2
item deleted at front-end is=10

1.Insert_front
2.Delete_front
3.Insert_rear
4.Delete_rear
5.display_list
6.Exit
Enter Choice:4
Item deleted at rear-end is 21
1.Insert_front
2.Delete_front
3.Insert_rear
4.Delete_rear
5.display_list
6.Exit
Enter Choice:4
Item deleted is 20

1.Insert_front
2.Delete_front
3.Insert_rear
4.Delete_rear
5.display_list
6.Exit
Enter Choice:4
List is Empty. Cannot delete!

1.Insert_front
2.Delete_front
3.Insert_rear
4.Delete_rear
5.display_list
6.Exit
Enter Choice:5
List empty. Cannot display items!

1.Insert_front
2.Delete_front
3.Insert_rear
4.Delete_rear
5.display_list
6.Exit
Enter Choice:7
Enter a Valid Choice!!
1.Insert_front
2.Delete_front
3.Insert_rear
4.Delete_rear
5.display_list
6.Exit
Enter Choice:6

Process finished.
```

L Lab Program -5 18/11/20 1BM19CS090

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. c) Display the contents of L-links d) delete the first element, specified element and last element in the list.

Code:

```
#include <stdio.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 ("Mem full \n");
    exit (0);
 }
 return x;
}
void freenode (NODE x)
{
  free (x);
}
NODE insert_rear (NODE first, int item)
{
   NODE temp, cur;
```

1BM19CS090
RAHIL

```
    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 Cannot 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 rear-end is %d", cur -> info);
  free (cur);
  prev -> link = NULL;
```

```
    return first;
}
NODE insert_pos (int item, int pos, NODE first)
{
    NODE temp, cur, prev;
    int count;
    temp = getnode ();
    temp -> info = item;
    temp -> link = NULL;
    if (first == NULL && pos == 1)
    {
        return temp;
    }
    if (first == NULL)
    {   printf ("Invalid Position\n");
        return first;
    }
    if (pos == 1)
    {
      temp -> link = first;
      first = temp;
      return temp;
    }
    count = 1;
    prev = NULL;
    cur = first;
    while (cur != NULL && count != pos)
    {
      prev = cur;
      cur = cur -> link;
      count ++;
    }
```

IBM19CS09
RAHIL

1BM19CS090
RAHIL

```
    if (count == pos)
    {
        prev -> link = temp;
        temp -> link = cur;
        return first;
    }
    printf ("Invalid position\n");
    return first;
}
NODE delete_pos (int pos, NODE first)
{
    NODE cur;
    NODE prev;
    int count, flag = 0;
    if (first == NULL || pos <0)
    {
        printf ("Invalid position \n");
        return NULL)
    }
    if (pos == 1)
    {
        cur = first;
        first = first -> link;
        free node (cur);
        return first;
    }
    prev = NULL;
    cur = first;
    count = 1;
    while (cur != NULL)
    {
        if (count == pos) { flag = 1; break; }
        count ++;
        prev = cur;
```

```
        cur = cur -> link;
    }
    if (flag == 0)
    {
        printf ("Invalid position \n");
        return first;
    }
    printf ("Item deleted at given position is %d \n", cur -> i
    prev -> link = cur -> link;
    freenode (cur);
    return first;
}
void display (NODE first)
{
    NODE temp;
    if (first == NULL)
    print ("list empty cannot display items\n");
    for (temp = first; temp != NULL ; temp = temp -> link)
    {
        printf ("%d\n", temp -> info);
    }
}
void main()
{
```

IBM19CS0
RAHIL