25-11-2020

Singly linked list :

```
# include <stdio.h>
# inculde <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("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 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 delete-pos (int pos, NODE first)
  {
    NODE prev, cur;
      int count;
     if (first == NULL || pos <= 0)
       {
       printf ("Invalid position\n");
          return NULL;
       }
    if (pos == 1)
    {
       cur = first;
       first = first->link;
       freenode (cur);
      return first;
    }
   prev = NULL;
   cur = first;
   count = 1;
```

```
while (cur != NULL)
  {
    if (count = = pos)
      {
        break;
      }
    prev = cur;
    cur = cur -> link;
    count++;
  }
 if (count ! = pos)
   { printf("Invalid position\n");
        return first;
   };
 prev -> link = cur -> link;
   ~~freefr~~
        freenode(cur);
        return first;
   }
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;
   }
```

```
int main()
{ int item, choice, pos;
    NODE first = NULL;
  for(;;)
printf(" \n1: Insert rear\n2: Delete front\n3:
       Delete-rear\n4: Display Position
       \n5. Exit);
 printf("Enter your choice\n");
 Scanf("%d", &choice);
  Switch (choice)
 {
Case 1: printf("Enter the item at rear-end\n");
        Scanf("%d", &item);
        first = insert_rear(first, item);
        break;

Case 2 : first = delete_front(first);
           break;

Case 3 : first = delete_rear(first);
           break;

Case 4 : printf("Enter the position\n");
         Scanf("%d", &pos);
         first = delete_pos(pos, first);
         break;
  default : exit(0);
  } break;
}
```