HEMANG SINGH

LAB - 7

1) Sort the linked list

```
#include <stdio.h>
#include <stdlib.h>
Struct node
  {
     int info;
     Struct node *link;
  };
typedef Struct node * NODE;
NODE  get node () {
      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_front (NODE first, int item) {
    NODE temp;
    temp = get node();
    temp -> info = item;
    temp -> link = NULL;
```

```
    if ( first == NULL)
        return 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 -> first 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 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;
    }
    prev->link = temp;
    temp->link = cur;
    return first;
}
NODE delete_info(int key, NODE first) {
    NODE prev, cur;
    if(first == NULL) {
        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(" key deleted is %d ", cur -> info);
freenode( cur);
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 -> link); }
}
```

```
Void main() {
    int item, choice, key;
    NODE first = NULL;
    for(;;) {
        printf("\n 1. Insert_front \n 2. Delete_front \n
                3. Insert_Rear \n 4. Delete_Rear \n
                5. Order_list \n 6. Delete_info \n
                7. Display_list \n 8. Exit \n");
        printf("\n Enter the Choice : ");
        scanf("%d", &choice);
        switch(choice) {
            Case 1: printf("\n Enter the Item at
                            front_end \n");
                    scanf("%d", &item);
                    first = insert_front(first, item);
                    break;
            Case 2: first = delete_front(first);
                    break;
            Case 3: printf("\n enter the item at
                            rear_end\n");
                    scanf("%d", &item);
                    first = insert_rear(first, item);
                    break;
            Case 4: first = delete_rear(first);
                    break;
```

```
case 5: printf("\n enter the item to be
                inserted in order_list\n");
        scanf("%d", &item);
        first = order_list(item, first);
        break;
case 6: printf("\n enter the key to
                be deleted \n");
        scanf("%d", &key);
        first = delek_info(key, first);
        break;
case 7: printf("\n");
        display(first);
        break;
default: exit(0);
        break;
}
}
}
```