Program for linked list

```
struct node {
    int info;
    ~~int~~ struct node *link;
};

typedef struct node *Node;

Node getNode(){
   Node x;

  x = (Node)malloc(sizeof(struct Node));
  if (x = NULL) {
    pf ("mem full \n");
    exit(0);
 }
  return x;
void free node (Node x){
    free(x);
}
Node insertFront(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 deleteFron (Node first){
      Node temp;
     if (first==NULL){
         printf("List is empty")};
       return first;
    }
    temp= first;
     temp =. temp->link
     pf(" Item deleted at front : %d", first->info);
     free(first);
     return temp;
}
```

```
Node insertRear ~~()~~ ( 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 = cur->link;
           cur -> link = temp;
   return first;
}

Node deleteRear (Node first){
    Node cur, prev;
    if ( first == NULL){
      printf ("List is empty\n");
      return first;
   }
  if ( first -> link == NULL ) {
  //  pf (" %d ", first -> info );
    free (first);
    retun NULL;
  }
```

```
prev= NULL;
 cur= first;
 while ( cur -> link != NULL ){
     prev =cur;
    cur  =  cur -> link;
   }
pf( "%d", cur->info );
  free (cur);
 Preve -> link  =NULL;
 return first;
```

```
NODE  delete_front (NODE first){
      NODE temp;
    if (first==NULL){
          pf("List is empty");
        return first;
    }
    temp = first
    temp= temp->link;
    Pf(" Item deleted at front end is=%d \n,
          first->info);
   free(first);
   return temp;
}
```

```
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) {
        preue = cur;
       cur = cur -> link;
 }
   prev -> lik => temp;
   temp -> link = cur;
   return first:
}
```

~~[illegible]~~ to bee

```
void display (Node first){
        Node temp;
        if (first == NULL){
                printf(" List is empty\n");
                return;
        }
    for (temp=first; temp!=NULL; temp=temp->link
                                          next)
            printf("%d", temp->info);
    }
printf("\n");

void main () {
    int item, choice, flag =1;
    Node first = NULL;
   while (flag ==1){
   printf("\n1. insert Front \n 2. insert Rear \n
           3. Delete Front \n 4. Delete Rear \n
            5. orderlist \n 6. Display \n 7. exit");
  printf ("enter your choice:");
  scanf ("%d", &choice);
```

```
switch(choice){
    case1:  printf("Enter the item to be inserted
                    at front :\n);
            scanf("%d", &item);
            first  = insert Front(first, item);
            break;
    case 2: printf("Enter the item to be inserted
                    at the rear : \n);
            scanf("%d", &item);
            first = insert Rear(first, item);
            break;
    case 3:  first = delete front(first);
             break;
    case 4:  first = delete Rear(first);
             break;
    case 5: printf("Enter element which will be placed
                    in order");
            scanf("%d", & item);
            first= orderlist(first, item);
            break;
    case 6 : desplay; break;
    case7 : exit(0)
}
```