Singcly linked list program —

```
#include<stdio.h>
# include <alloc.h>
# include<process.h>
 struct node
{ int info;
    struct node *link;
};
typedef struct node *NODE;
 NODE getnode()
{ NODE = x
```

```
x = (NODE)malloc (size of (struct node));
if (x==NULL)
{ printf ("mem full\n");
  exit(0);
}
return x; /*creates a node for me and return the
                                                    address*/
}
void freenode(NODE x)
{ free(x);
}
NODE insert_front(NODE first, int item)
{ NODE temp;
  temp = getnode(); /* obtain the node from available list*/
  temp->info = item; /*Insert the item in new node created*/
  temp->link = NULL;
  if (first == NULL) /* checking for my list is empty*/
  return temp; /* if my list is empty the temp node
                            will be first nod */
  temp->link = first; /*attach the new node at front
                           end of list */
  first = temp;
  return first;         /* Address of first will be assigned to
                            link field of temp*/
}
NODE delete_front (NODE first)
{
  NODE temp;
  if (first == NULL)
{
  printf("List is empty cannot delete\n");
  return first;
```

Scanned with CamScanner

```
temp = first;
temp = temp -> link;
printf("item deleted at front-end is =%d", first->i
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("item deleted is %d \n", first->info);~~
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 rear - end is %d", cur -> info);
free (cur);
prev -> link = NULL;
return first;
}
```

```
void display (NODE first)
{ NODE temp;
 if (first == NULL)
 pf("list empty can't display items\n");
 for(temp=first; temp!=NULL; temp=temp->link)
 {
   pf ("%d\n", temp -> info);
 }
}
```

Scanned with CamScanner

```
void main ()
{ int item, choice, pos;
  NODE first = NULL;
  for (; ;)
  { printf("\n1: Insert_front ~~[illegible]~~ \n2: Delete_front\n 3:Insert_rear
          \n4: Delete_rear\n 5: Insert_pos\n
          6: Display_list \n 7: Exit\n");
    pf(" Enter choice");
    scanf("%d\n", &choice);
    switch(choice)
    {
     case 1: pf("Enter the item at front-end\n");
             scanf("%d", &item);
             ~~[illegible]~~
     ~~[illegible]~~  first = insert ~~Delete~~ _front(first, item);
              break;

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

     case 3:  pf("Enter item at rear_end\n");
             scanf("%d", &item);
             first = insert_rear (first, item);
              break;
     case 4: first = delete_rear (first);
            break;
     case 5; display (first);
             break;
     ~~case 6:~~
       default : exit(0); break; } } }
```