Singly linked list

```
#include <stdio.h>
#include <conio.h>
#include <alloc.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)
{
  pf("mem full\n");
  exit(0);
}
  return x;
}
void freenode(NODE x)
{
  free(x);
}
NODE insert_front(NODE first, int item)
{
NODE temp;
temp = getnode(); // obtain the node from avail list
temp -> info = item; // insert item in new node created.
temp -> link = NULL; //
```

```
if (first == NULL)
    return temp;
temp -> link = first;
    first = temp;
    return first;
}
NODE delete_front (NODE first)
{
  NODE first, temp;
  if (first == NULL)
    pf ("list is empty cannot delete");
    return first;
  }
  temp = first;
  temp = temp -> link;
pf ("item deleted at front-end is = %d\n", first->
  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)
    {
        pf ("list is empty");
        return first;
    }
    if (first -> link == NULL)
    {
        pf ("item deleted is %d\n", first -> info);
        free (first);
        return NULL;
    }
    prev = NULL;
    cur = first;
    while (cur -> link != NULL)
    {
        prev = cur;
        cur = cur -> link;
    }
    pf ("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 \n");
for (temp = first; temp != NULL; temp = temp -> link)
   {
    pf ("%d \n", temp -> info);
   }
 }
void main ()
 {
  int item, choice, pos;
   NODE first = NULL;
   for (;;)
pf (" 1. Insert front 2. Delete front 3. Insert rear
  4. Delete rear 5. insert pos 6. display list
 7. Exit");
   pf (" Enter choice ");
     sf ("%d", &choice);
  switch (choice)
```

```
}
case 1: printf(" Enter item at front-end \n");
        scanf("%d", &item);
        first = insert front (first, item);
        break
case 2: first = delete front (first);
        break;
case 3: printf(" Enter item at rear-end");
        scanf("%d", &item);
        first = insert rear (first, item);
        break;
case 4: first = delete_rear (first);
```

```
case 6: display(free);
        break;
case 7: exit(0);
        break;
    }
}
getch();
}
```