Lab - 5
Linked list

```
#include <stdio.h>
#include <stdlib.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;
}
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 = 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->link != NULL)
{ cur = cur -> Link; }
cur ->
cur -> link = temp;
return first;
}
NODE delete_rear(NODE first)
{
   NODE cur, prev;
   if( first == NULL)
{
   printf (" list is empty \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 : %d \n",
        , cur -> info );
free(cur);
prev -> link = NULL;
return first;
}
void Display (NODE first)
{
   NODE temp;
   if (first == NULL)
{ printf(" list empty \n") }
for (temp = first ; temp != NULL; temp
                                = temp -> link)
{
   printf("%d \n", temp -> info);
}
}
```

```
void main()
{
  int item, choice;
  NODE first = NULL;
  for(;;)
{
printf("\n 1: Insert_front \n 2: Delete_front
        \n 3: Insert_rear \n 4: Delete_rear
        \n 5: Display \n 6: Exit \n");
printf("Enter choice \n");
scanf("%d", &choice);
switch(choice)
{
case 1: printf("Enter item \n");
        scanf("%d", &item)
break; first = Insert_front(first, item);
break;
case 2: first = delete_front(first);
break;
case 3: printf("Enter item \n");
        scanf("%d", &item);
        first = insert_rear(first, item);
break;
```

```
case 4 : first = delete_rear (first);
break;
case 5 : display (first);
break;
default : exit (0);
break;
}
}
}
```