Date ___/___/_____

Q Implementing stack using singly 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 ( 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 = getnode ();
    temp->info = item;
    temp ->link = NULL;
```

Page No.

Date ___ / ___ / ______

```
if(first = = NULL)
Return temp;
temp -> link = first;
 first = temp;
 Return first;
}
NODE delete_front(NODE first)
 {
   NODE temp;
  if(first = = NULL)
   {
     printf("stack is ~~[illegible]~~ empty cannot delete \n");
     return first;
   }
temp = first;
temp = temp -> link;
printf(" item deleted at frontend is = %d \n",
                                  first -> info);
free(first);
Return temp;
}
void display(NODE first)
 {
  NODE temp;
  if (first = = NULL)
  printf(" stack empty cannot display items \n");
  for (temp = first; temp != NULL; temp = temp ->
                                              link)
  {
   printf("%d \n", temp -> info);
  }
}
```

Date ___/___/___

```
int main()
{
 int item, choice, pos;
 NODE first =NULL;
  for(;;)
 {
 printf("\n 1: Insert front \n 2: Delete-front \n
             3: Display-list 4: Exit \n");
printf("enter the choice \n");
scanf("%d", &choice);
printf("-----\n");
switch (choice)
{
 case 1: printf("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: display (first);
 break;
 default: exit(0);
 break;
 }
 }
}
```

Date __/__/____

Q Implement ~~using~~ queue using linked list

```
# ~~struct~~ include <stdio.h>
# include <stdlib.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)
    {
        printf ("memory full \n");
        exit(0);
    }
    return x;
}
void freenode (NODE x)
{
    free(x);
}
NODE insert_rear (NODE first, int item)
{
        NODE temp, cur;
        temp = getnode ();
        temp-> info= item;
        temp -> link =NULL;
```

Page No.

Date ___ /___ /______

```
if (first == NULL)
return temp;
cur = first;
while (cur -> link != NULL)
 cur = cur -> link;
 cur -> link = 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;
 }
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->info);
    }
```

Page No.

Date ___/___/___

```
int main ()
{
 int item, choice, pos;
 NODE first = NULL;
 for (;;)
 {
    printf ("\n 1: InsertRear \n 2. Delete_front
            \n 3. Display_list \n 4. Exit \n");

printf ("enter the choice\n");
scanf ("%d", &choice);
 printf ("----\n");
 switch (choice)
 {
  case 1: printf ("enter the item at rear-end \n");
  scanf ("%d", &item);
  first = insert_rear (first, item);
   break;
 case 2: first = delete_front (first);
  break;
   default: exit(0);
   break;
 }
 }
}
```