Implementation of stacks & Queues using link lists.

1BM19EE025

```
#include <stdio.h>
#include <stdlib.h>
 struct node
{
   int info;
   struct node * link;
 };
typedef. struct node *NODE;

NODE first = NULL;
NODE second = NULL;
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 = 
temp->link = NU
 if (first == NU
    return temp
  cur = first;
  while(cur->li
     cur = cur -
   cur->link =
   return first
}

NODE insert_r(N
{
   NODE temp, cu
   temp = getnod
   temp->info =
    temp->link =
     if (second =
        return
    cur = second
    while (cur-
       cur = cur
    cur->link
    return seco
}
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 insert_r(NODE second, int item)
{
    NODE temp, cur;
    temp = getnode();
    temp->info = item;
    temp->link = NULL;
    if (second == NULL)
        return temp;
    cur = second;
    while (cur->link != NULL)
        cur = cur->link;
    cur->link = temp;
    return second;
}
```

```
NODE delete_front (NODE second)
{
  NODE temp;
  if(second == NULL)
  {
   printf(" Queue is empty cannot delete\n");
   return second;
  }
  temp = second;
  temp = temp->link;
  printf(" Item deleted is = %d\n", second->info);
  free(second);
  return temp;
}
NODE delete_rear(NODE first)
{
   NODE cur, prev;
   if(first == NULL)
   {
    printf(" Stack is empty cannot 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 -> link = NULL~~;
        prev = cur;
        cur = cur -> link;
}
printf ("Item deleted is %d \n", cur->info);

free (cur);
prev -> link = NULL;
return first;
}
void display (NODE first)
{
    ~~tode~~ 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);
}
void disp (NODE second)
{
  NODE temp;
  if(second == NULL)
    printf ("queue empty cannot display items \n");
  for( temp = second; temp != NULL; temp = temp -> link)
    printf (" %d \n", temp -> info);
```

```
int main()
{
    int item, choice, f;
    for(;;)
    { printf("\n1. Stack \n2. Queue \n3. exit\n");
      scanf("%d", &f);
      if (f==1)
      {
        printf("\n stack :- ");
        for(;;)
        {
          printf("\n1. Push \n2. Pop \n3. Display \n4. Return to
          Main menu \n5. Exit \n");
          printf("Enter the choice :");
          scanf("%d", & choice);
          switch (choice)
          {
                Case 1: printf ("Enter the item: ");
                        scanf ("%d", & item);
                        first = insert_rear(first, item);
                        break;
                Case 2: first = delete_rear(first);
                        break;
                Case 3: printf (" stack : \n");
                        display (first);
                        break;
                case 4: main();
                Case 5: exit(0);
default : printf ("Enter valid i/p \n");
break;

}

}

}
if (f==2){
   printf ("\n Queue :- ");
   for(;;){
   printf ("\n1. Push \n2. Pop \n3. Display \n4. Return to
           Main Menu \n5. Exit \n");
    printf ("Enter the choice \n");
     scanf ("%d", &choice);
    switch (choice)
   {
        case 1 : printf ("Enter the item: ");
             scanf ("%d", &item);
             second = insert_r (second, item);
             break;
       case 2: second = delete_front (second);
             break;
       case 3: printf ("Queue: \n");
            disp (second);
           break;
       Case 4 : main();
      Case 5 : exit(0);
      default : printf ("Enter correct instruction \n");
          break;
  }
}}
```

```
        else if (f==3)
        {  exit(0);
        }
        else {
            printf ("Enter proper choice...");
          }
        }
        return 0;
    }.
```