9|12|2020 DS-LAB-PROGRAM

PROGRAM-1 (Doubly Linked List)

~~WAP to implement stack and queues using linked representation~~

CODE:-

```
#include <stdio.h>
#include <stdlib.h>
#include <process.h>
struct node
 {
  int info;
  struct node* rlink;
  struct node* llink;
 };
 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 temp, cur;
  temp = getnode ();
```

```
temp->rlink = NULL;
temp->llink= NULL;
temp-> info= item;
cur = head ->llink;
temp->llink = NULL; cur;
cur-> rlink = cur;
cur ->rlink = temp;
head ->llink =temp;
temp ->rlink = head;
head ->info = head ->info +1;
return head;

}
NODE insert_leftpos (int item, NODE head)
{
 NODE temp, cur, prev;
  if (head->rlink ==head)
  {
   printf ("list empty\n");
    return head;
   }
  cur= head ->rlink;
  while (cur != head)
  {
  if (item== cur ->info) break;
   cur= cur ->rlink;
 }
 if (cur==head)
 {
  printf( "key not found \n");
  return head;
 }
 prev =cur->llink;
```

Scanned with CamScanner

```
1] printf ("enter towards left of %d = ", item);
   temp = getnode ();
   scanf ("%d", &temp->info);
    prev->rlink = temp;
    temp->llink = prev;
     cur->llink = temp;
     temp->rlink = cur;
     return head;
    }

    NODE insert_rightpos (int item, NODE head)
    {
     NODE temp, cur, prev;
     if (head->rlink == head)
     {
      printf ("list empty \n");
      return head;
     }
     cur = head->rlink;
     while (cur != head)
     {
      if (item == cur->info) break;
      cur = cur->rlink;
      }
      if (cur == head)
      {
       printf ("key not found \n");
       return head;
     }
      prev = cur->rlink;
      printf ("enter towards left of %d = ", item");
      temp = getnode();
```

```
1) scanf ("%d", &temp->info);
   prev->llink = temp;
   temp->llink = cur;
   cur->rlink = temp;
   temp->rlink = prev;
   return head;
   }
   NODE delete_all_key(int item, NODE head)
   {
    NODE prev, cur, next;
    int count;
     if (head->rlink == head)
     {
      printf ("LE");
      return head;
      }
      count = = 0;
      cur = head->rlink;
       while (cur != head)
      {
       if (item != cur->info)
       cur = cur->rlink;
        else
       {
       count ++;
       prev = cur->llink;
       next = cur->rlink;
       prev->rlink = next;
       next->llink = prev;
       freenode (cur);
        cur = next;
       }
     }
```

1]
```
if (count == 0)
printf ("key not found");
else
printf ("key found at %d positions and are deleted \n", count

 return head;

}
 void search - info (int item, NODE head){
  NODE cur;
  if (head -> rlink == head)
  {
   printf ("list empty \n");
  }
  cur = head -> rlink;
  while (cur != head)
  {
   if (item == cur -> info)
    {
     printf (" Search Succesfully \n");
     break;
    }
    cur = cur -> rlink;
    }
    if (cur == head)
    {
     printf (" info not found \n");
    }
   }

   void display (NODE head)
```

```
1] {
    NODE temp;
    if (head->rlink==head)
    {
     printf("list empty\n");
     return;
    }
    for (temp=head->rlink; temp!
       =head; temp=temp->rlink)
      printf("%d\n", temp->info);
    }
    void main()
   {
    int item, choice, key;
     NODE head;
     head= getnode();
     head->rlink =head;
     head->llink= head;
     for (;;)
    {
     printf("\n1.insert_rear\n2.insert-key-fet left\n3.insert_key-
              -right\n4.delete-duplicates\n5.Search-info\n6.display
              \n7.exit\n");
         Scanf("%d", &choice);
          Switch (choice)
         {
          case1: printf("enter the item\n");
           Scanf("%d", &item);
           head = insert-rear (head, item);
           break;
```

1]

```
case 2: printf("enter the key item\n");
  scanf("%d", &item);
  head = insert-leftpos(item, head);
  break;
case 3: printf("enter the key item\n");
  scanf("%d", &item);
  head = insert_rightpos (item, head);
  break;
case 4: printf("enter the key item\n");
  scanf("%d", &item);
  head = delete- all-key (item, head);
  break;
case 5: printf("enter the key item\n");
  scanf("%d", &item);
  search-info(item, head);
  break;
case 6: display (head);
  break;
  defaut : exit(0);
  break;
}
}
}
```

PROGRAM-2

WAP to implement stack & queues using linked representation :-

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
struct node
{
 int data;
 struct node* next;
 };
 typedef struct node *NODE;
 NODE get()
 {
  NODE x;
  x =(NODE) malloc (sizeof (struct node));
  if(x==NULL)
  {
   printf ("Memory full\n");
    exit (0);

  }
   return x;
  }
  void f(NODE X)
 {
   free(x);
  }
  NODE insert-front (NODE first, int item)
   {
    NODE temp;
    temp=get();
```

Scanned with CamScanner

2]

```
temp->data = item;
temp->next = NULL;
if (first == NULL)
{
  return temp;
}
 temp->next = first;
  first = temp;
  return first;
}
NODE delete_front (NODE first)
{
   NODE temp;
   if (first == NULL)
   {
    printf ("List Empty\n");
     return first;
   }
   temp = first;
   temp = temp->next;
   printf ("%d is deleted ", first->data);
   f (first);
    return temp;
}
NODE insert_rear (NODE first, int item).
{
   NODE temp, cur;
   temp = get();
   temp->data = item;
   temp->next = NULL;
   if (first == NULL)
```

Scanned with CamScanner

```
 {
  return temp;
 }
 cur = first;
 while (cur -> next != NULL)
 {
   cur = cur -> next;
   cur -> next = temp;
 }
 return first;
}
NODE delete_rear (NODE first)
{
  NODE cur, prev;
  if (first == NULL)
   {
     printf (" List Empty \n");
     return first;
   }
   if (first -> next == NULL)
   {
     printf ("%d is deleted \n", first->data);
     f(first);
     return NULL;
   }
   prev = NULL;
   cur = first;
   while (cur -> next != NULL)
   {
     prev = cur;
     cur = cur -> next;
   }
```

2]

```
printf ( "item deleted = %d \n", cur->data);
f (cur);
prev->next = NULL;
return first;
}
void display (NODE first)
{
   NODE temp;
   if (first == NULL)
   {
    printf ( "List Empty \n");
   }
   for (temp = first ; temp != NULL ; temp = temp->next)
   {
    printf ("%d \n", temp->data);
    }
   }
   int main()
   {
    int item, ch;
    NODE first = NULL;
     do {
      printf (" 1. Insert front \n2. Delete front \n3. Insert rear
                 \n4. Delete rear \n5. Display \n6. Exit \n");

       printf ("Enter choice :");
        scanf ("%d", &ch)
       switch (ch)
        {
         case 1:
```

Scanned with CamScanner

2)
```
printf ("Enter item at front:");
scanf ("%d", &item);
 first = insert- front (first, item);
 break;
 case2:
  first = delete- front (first);
  break;
 case 3:
 printf ("Enter item at rear:").
 scanf ("%d", &item);
 first = insert- rear (first, item);
  break;
  case 4:
   first = delete- rear (first);
   break;
   case5:
   display (first);
   break;
   case 6:
   break;
   default:
    printf ("\nEnter a valid choice\n");
    break;
  }
 }while (ch!=6);
 return0;
}
```