Date __/__/____

Q. WAP to implement singly linked list with following operation :-

a) Create a linked list
b) Insertion of a node at first position, at any position and at end of list
c) Deletion of first element, specified element and last element in the list
d) Display the content of the linked list.

```
# insert <stdio.h>
# insert<stdlib.h>
# include <conio.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 ("memory full\n");
    exit (0);
    }
    return x;
}
```

Date __/__/____

```
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 -> link = first;
  first = temp;
  return first;
}
NODE delete_front (NODE first)
{
   NODE temp;
   if (first = = NULL)
   {
     printf ("list is empty cannot delete \n");
    return first;
   }
  temp = first;
  ~~if (first !== NULL)~~
  temp = temp -> link;
  printf ("item deleted at front end is = %d \n"
                                          , first -> info);
  free (first);
 return temp;
}
```

Scanned with CamScanner

```
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)
     {
       printf("list is empty cannot delete \n");
       return first;
     }
   if(first -> link == NUL)
   {
     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 is %d", cur->info);
    free(cur);
    prev->link = NULL;
    return first;
  }
  NODE delete_pos(int pos, NODE first)
  {
    NODE prev, cur;
    int count;
    if(first == NULL || pos <= 0)
    {
      printf("Invalid position \n");
      return NULL;
    }
    if(pos == 1)
    {
      cur = first;
      first = first->link;
      freenode(cur);
      return first;
    }
    prev = NULL;
    cur = first;
    count = 1;
    while(cur != NULL)
    {
      if (count == pos)
      {
```

Page No.

Scanned with CamScanner

Date __/__/__

```
    } break;
    }
   prev = cur;
   cur = cur -> link;
    count ++;
  }
  if (count != pos)
  {
   printf("Invalid position \n");
    return first;
  }
 prev -> link = cur -> link;
 freenode (cur);
 return first;
}
NODE Insert_pos (int item, int pos, NODE first)
{
  NODE temp, cur, prev;
  int count;
 temp = getnode();
 temp -> info = item;
 temp -> link = NULL;
 if (first == NULL && pos == 1)
 {
  return temp;
 }
 if (first == NULL)
 {
  printf("Invalid position \n");
  return NULL;
}
```

Scanned with CamScanner

Date __/__/__

```
if(pos==1)
{
   temp->link = first;
   return temp;
}
count = 1;
prev = NULL;
cur = first;
while(cur != NULL && count != pos)
{
  prev = cur;
  cur = cur ->link;
  count++;
}
if (count == pos)
 {
   prev ->link = temp;
   temp -> link = cur;
   return first;
 }
printf("Invalid position position \n");
return first;
}
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);
   }
}
```

Scanned with CamScanner

```
void main()
{
 int item, choice, pos;
 NODE first = NULL;
 for(;;)
 {
  printf("\n 1: Insert_front \n 2: Delete front \n
  3: Insert_rear \n 4. Delete ~~at spec~~ rear \n 5. Delete
  at specified position \n 6: Insert at specified position
 \n 7: Display");
 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 : printf ("enter the item at rear end \n");
   scanf ("%d", &item);
   first = insert_rear(first, item);
   break;
 Case 4 : first = delete_rear(first);
  break;
  Case 5 : printf ("Enter the position:\n");
  Scanf ("%d", & pos);
 Case 6 : printf ("Enter the item and the position
                                              : \n");
 Scanf ("%d", & item, &pos);
```

Date ___/___/______

```
first = insert_pos (item ,pos , first);
break;
case 7: display( first);
break;
default : exit(0);
 break;
}
 }
}
```

Scanned with CamScanner