HEMANG SINGH
1BM19CS061

LAB-5
&
LAB-6

1) 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_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  = = 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 is %d",
                                cur -> info);
free (cur);
prev -> link = NULL;
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 first;
}
```

```
if (pos ==1){
      temp -> link = first;
      first = temp;
      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");
 return first;
}
NODE delete_pos (int pos, NODE first) {
        NODE cur;
        NODE prev;
        int count ,flag = 0;
```

```
if ( first == NULL || pos<0) {
        printf(" Invalid position ");
        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 ){
        flag = 1;
        break;
}
count ++;
prev = cur;
cur = cur -> link;
}
if( flag = =0) {
    printf(" Invalid position \n");
    return first;
}
```

```
printf(" Item deleted at given position is %d \n",
                                        cur->info);

  prev -> link = cur-> link;
  freenode (cur);
   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 );

  }

}

void   main()
{
  int  item , choice , key , pos ;
  int   count = 0;
  NODE first  = NULL;
  for( ;;) {
     printf(" \n 1. Insert_rear \n 2. Delete_rear \n
                      3. Insert - info- position \n 4. Delete_info
                       position \n 5. Display_list \n 6. Exit\n");
     printf(" Enter the choice \n");
     scanf(" %d", & choice \n");
     switch( choice) {
```

```
~~case 1: printf("Enter the item at rear-end");~~
~~        scanf("%d", &item);~~
~~        printf("Enter the position\n");~~
~~        scanf("%d", &pos);~~
~~        first ~~&~~ = insent~~

~~Case 1: printf("Enter the item to be~~
~~                 inserted at given position~~
~~        scanf("%d", & item);~~

        pri

Case 1: printf("Enter the item at rear-
                                   end\n");
        scanf("%d", & item);
        first = insert_rear(first, item);
        break;
Case 2: first = delete_rear(first);
        break;
Case 3: printf("Enter the item to be
                   inserted at given position\n");
        Scanf("%d", & item);
        printf("Enter the position\n");
        Scanf("%d", & pos);
        first = insent_pos(item, pos, first);
        break;
Case 4 :  printf ("Enter the position \n");
          Scanf (" %d ", &pos);
          first = delete_pos( pos , first) ;
          break;
   Case 5 : display ( first ) ;
             break;
   default :  exit( 0 );
              break;
 }
 }
}
```