Date ___/___/___

Q WAP Implement Single link list with following operations
a) Sort the linked list
b) Reverse the linked list
c) Concatenation of two linked lists.

```
# include<stdio.h>
# include <stdlib.h>
# include <conio.h>
structnode
 {
    int info;
    struct node * link;
 };
typedef struct node* NODE;
NODE getnode()
{
  NODE ~~getnode~~ & x;
  x =(NODE) malloc(size of (struct node));
  if (x == NULL)
   {
      printf("memory full\n");
      exit(0);
   }
   return x;
 }
void freenode(NODE x)
 {
  free(x);
 }
```

Scanned with CamScanner

Date __/__/__

```
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;
 temp = temp -> link;
 printf(" item deleted at front end is = %d \n",
                                  first -> info);
 free(first);
 return temp;
}
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 != NULL && item > cur info~~
  {      cur -> link != NULL)
  prev = cur;
   cur = cur -> link;
}
printf (" item deleted at rear end is %d", cur -> info)
free (cur);
```

Date ___/___/___

```
~~temp -> link = cur;~~   prev -> link = NULL;
 return first;
}
NODE ~~order~~ ~~(NODE first~~ order_list(int item, NODE first)
{
 NODE temp, prev, cur;
 temp = getnode();
 temp -> info = item;
 temp -> link = NULL;
 if (first == NULL) return temp;
 if (item < first -> info)
 {
   temp -> link = first;
   return temp;
 }
 prev = NULL;
 cur = first;
 while (cur != NULL && item > cur -> info)
 {
   prev = cur;
   cur = cur -> link;
 }
 prev -> link = temp;
 temp -> link = cur;
 return first;
}
NODE reverse (NODE first)
{
   NODE cur, temp;
   cur = NULL;
   while (first != NULL)
 {
```

Page No.

```
~~printf ("item deleted at rear end~~

    temp = first;
    first = first -> link;
    temp -> link = cur;
    cur = temp;
  }
 return cur;
}
NODE concat ( NODE first, NODE second)
{
    NODEcur;
  if (first = = NULL)
   return second;
  if (second = = NULL)
  return first;
  cur = first;
  while (cur -> link != NULL)
  cur = cur -> link;
  cur -> link = second;
  return first;
}
void display (NODE first)
{
  NODE temp;
  if ( first == NULL)
 printf ("list empty cannot diplay items \n");
 for (temp = first ; temp != NULL; temp = temp->
                                         link)
{
 printf ("%d\n", temp -> info) ;
```

Date ___/___/___

```
    }
}
}
void main()
{
 int item, choice, n, i;
 NODE first = NULL, a, b;
 for(;;)
{
  printf("\n 1: Insertfront \n 2: Delete_front &\n
  3: Insert_rear \n 4: Delete_rear \n 5: Order_
list \n 6: reverse_list \n 7: Concat_list \n
8: Display_list \n 9: 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: 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 item tobe inserted
                in ordered_list \n");
```

Scanned with CamScanner

Date __/__/____

```
scanf ("%d", &item);
first = orderlist (item, first);
break;
case 6: first = reverse (first);
display (first);
break;
Case 7: printf ("Enter the no. of nodes in 1 \n");
 scanf ("%d", &n);
 a = NULL;
 for (i = 0; i < n; i++)
 {
    printf ("Enter the item \n");
    scanf ("%d", &item);
    a = insert_rear (a, item);
 }
 printf ("Enter the no. of nodes in 2 \n");
 scanf ("%d", &n);
 b = NULL;
 for (i=0; i<n; i++)
 {
   printf ("Enter the item \n");
   scanf ("%d". , & item);
  b = insert_rear (b, item);
 }
 a = concat (a, b);
 display (a);
break;
Case 8: display (first);
 break;
default: init(0);
break;
}
```