10/12/20.

DISHA B.
1BM21CS050.

```
void insertAtEnd(struct node **head, int d)
{
  struct node *temp, *n;
  if (*head == NULL)
   {  temp = (struct node**) malloc(sizeof(struct
                                    node));
      temp -> data = d;
      temp -> next = NULL;
    *head = temp;
   }
  else {
       temp = *head;
       // go to the last node
       while(temp->next != NULL)
         {
           temp = temp->next;
         }
       // adding node at the end
       n = (struct node*) malloc(sizeof(struct
                                    node));
       n->data = d;
       n-> next = NULL;
       temp->next = n;
      }
  }
void reverse(struct node **head)
{ struct node *prev, *cur, *next;
  cur = *head;
  prev = NULL;
  next = NULL;
```


Disha

Disha B
1BM19CS088.

```
if(*head == NULL)
{
    printf("Empty LIST\n");
    return;
}
while(cur != NULL)
{
    next1 = cur->next;
    cur->next = prev;
    prev = cur;
    cur = next1;
}
*head = prev;
}
void concat(struct node *head1, struct node *head2)
{
  if(*head1 == NULL)
    { *head1 = *head2;
      return;
    }
    if(*head2 == NULL)
    { *head2 = *head1;
      return;
    }
  struct node *temp = *head1;
  while(temp->next != NULL)
    { temp = temp->next;
    }
  temp->next = *head2;
}
```

Disha B
1BM19CS050.

```
struct node* merger(struct node* a, struct node* b)
{
    // base case
    if (a == NULL)
    {
        return b;
    }
    if (b == NULL)
    {
        return a;
    }
    struct node* c = NULL;
    // rec case
    if (a->data < b->data)
    {
        c = a;
        c->next = merger(a->next, b);
    }
    else {
        c = b;
        c->next = merger(a, b->next);
    }
    return c;
}
struct node* MidPoint(struct node* head)
{
    if (head == NULL || head->next == NULL)
    {
        return head;
    }
    struct node* fast = head->next;
    struct node* slow = head;
```

```
while(fast != NULL && fast->next != NULL)
{
    fast = fast->next->next;
    slow = slow->next;
}
return slow;
}
struct node *MergeSort(struct node *head)
{
if(head == NULL || head->next == NULL)
{
    return head;
}
// rec case
// 1. Breaking into 2
struct node *mid = midPoint(head);
struct node *a = head;
struct node *b = mid->next;
mid->next = NULL;
// 2. rec sort the two parts
a = MergeSort(a);
b = MergeSort(b);
// 3. Merging them.
struct node *c = merge(a,b);
return c;
}
void display(struct node *head)
{ while(head != NULL)
{
```

```
    printf("%d->", head->data);
    head = head->next;
  }
  printf("\n");
}
int main()
{
  struct node
  *head1 = NULL, *head2 = NULL, *head3 = NULL,
  *head4 = NULL, *ans = NULL;
  int data, n;
  printf("---- SORTING ----\n");
  printf("Enter the list to be sorted (Enter -1 to
    stop):\n");
  scanf("%d", &data);
  while(data != -1)
  {
    insertAtEnd(&head1, data);
    scanf("%d", &data);
  }
  printf("List before sorting:");
  display(head1);
  ans = mergeSort(head1);
  printf("List after sorting:");
  display(ans);
  printf("\n---- REVERSE ----\n");
  printf("Enter the list to be reversed (Enter
    -1 to stop):\n");
  scanf("%d", &data);
  while(data != -1)
  {
```

Disha.B
1BM19CS050.

```
    insertAtEnd(&head2,data);
    scanf("%d",&data);
}
printf("List before reversing:");
display(head2);
reverse(&head2);
printf("List after reversing:");
display(head2);
printf("\n---CONCATENATION---\n");
printf("Enter the first list(Enter -1 to stop)\n");
scanf("%d",&data);
while(data!=-1)
{
    insertAtEnd(&head3,data);
    scanf("%d",&data);
}
printf("Enter the second list(Enter -1 to stop)\n");
scanf("%d",&data);
while(data!=-1)
{
    insertAtEnd(&head4,data);
    scanf("%d",&data);
}
printf("First list:");
display(head3);
printf("Second list:");
display(head4);
concat(&head3,&head4);
printf("Concatenated list:");
display(head3);
return 0;
}
```