1BM19CS167
SWAROOP. S. JADHAV

Lab → 7 ↳ Sort

```
void Sort (struct Node * start)
{
    int swapped, i;
    struct Node *ptr1;
    struct Node *lptr = NULL;
    if (start == NULL)
        return;

    do
    {   swapped = 0;
        ptr1 = start;

        while (ptr1->data > ptr1->next->data)
        {
            swap (ptr1, ptr1->next);
            swapped = 1;
        }
        ptr1 = ptr1->next;
    }
    lptr = ptr1;
  }
  while (swapped);
}

void swap (struct Node *a, struct Node *b)
{
    int temp = a->data;
    a->data = b->data;
    b->data = temp;
}
```

1BM19CS167
SWAROOP. S. JADHAV

2. REVERSE

→
```
void Reverse (struct Node **head_ref)
{
    struct Node * prev = NULL
    struct Node * current = *head_ref;
    struct Node * next = NULL;
    while (current != NULL)
    {
        next = current->next;
        current->next = prev;
        prev = current;
        current = next;
    }
    *head_ref = prev;
}
```

3. CONCATENATE

→
```
void concatenate(struct node *a, struct node *b)
{
    if (a!=NULL && b!=NULL)
    {
        if (a->next == NULL)
            a->next=b;
        else
            concatenate (a->next, b);
    }
    else
    {
        printf("Either a or b is NULL\n");
    }
}
```

1BM19CS167

SWAROOP S JADHAV

```
struct node *concat (struct node * start1, struct node * start2)
{
    struct node *ptr;
    if (start1==NULL)
    {
        start1=start2;
        return start1;
    }
    if (start2==NULL)
        return start1;
    ptr=start1;
    while (ptr->link!=NULL)
        ptr=ptr->link;
    ptr->link=start2;
    return start1;
```

→ 4. STACK IMPLEMENTATION

```
void push (struct Node **head_ref, int new_data)
{
    struct Node * new_node=(struct Node*) malloc (sizeof (struct Node));
    new_node->data=new_data;
    new_node->next=(*head_ref);
    (*head_ref)=new_node;
}

void pop()
{
    struct node *ptr
    if (head==NULL)
```

```
    printf("List is empty");
}
else
{
    ptr = head;
    head = ptr->next;
    free(ptr);
    printf("Node deleted from the beginning");
```

→ QUEUE IMPLEMENTATION

```
void Enqueue(item)
{
    struct Node *ptr, *temp;
    ptr = (struct Node*) malloc(sizeof(struct Node));
    ptr->data = item;
    ptr->next = NULL;
    if(head==NULL)
    {
        head = ptr;
        printf("Node inserted");
    }
    else
    {
        temp = head;
        while(temp->next != NULL)
        {
            temp = temp->next;
        }
        temp->next = ptr;
        printf("Node inserted");
    }
}
```

1BM19CS167
SWAROOP. S. JADHAV

```
void Dequeue()
{
  struct Node * ptr;
  if (head==NULL)
  {
    printf("List is empty");
  }
  else
  {
    ptr=head;
    head = ptr->next;
    free(ptr);
  printf("Node deleted from the begining");
  }
}
```