1. Given an unsorted linked list, and without using a temporary buffer, fill in the blanks in the below method that will delete any duplicates from the linked list. (Brute force solution is acceptable)

```
void RemoveDuplicates (Node * head)
{
        Node * temp1, * temp2, * prev;
        temp1=temp2=prev=head;
        while( _____________________________)
        {
                while(_____________________________)
                {
                        if(temp1->data == temp2->data)
                        {
                                //delete
                                _______________ = _______________;
                                free(temp2);
                        }
                        prev = temp2;
                        temp2=temp2->next;
                }
                temp1=temp1->next;
        }
}
```

Answer:

```
void RemoveDuplicates (Node * head)
{
        Node * temp1, * temp2, * prev;
        temp1=temp2=prev=head;
        while( temp1!=NULL)
        {
                while( temp2!=NULL)
                {
                        if(temp1->data == temp2->data)
                        {
                                //delete
                                prev ->next = temp2->next;
                                free(temp2);
                        }
                        prev = temp2;
                        temp2=temp2->next;
                }
                temp1=temp1->next;
        }
}
```

2. Given two sorted linked lists, merge the lists to form a single sorted linked list. Reuse one of the lists, Complete the given code below.

```
Node * Merge (Node * list1, Node * list2) {
        if (list1 == null) return list2;
        if (list2 == null) return list1;

        Node * newhead,temp;
        if (list1->data < list2->data) {
                newhead = list1;
                list1= list1->next;
        }
        else {
                newhead = list2;
                list2 = list2->next;
        }
        temp = newhead;

        while(_____________________) {
                if (__________ > __________){
                        temp->next = list2;
                        list2 = list2->next;
                }
                else {
                        temp->next = list1;
                        list1 = list1->next;
                }
                temp = temp->next;
        }
        if (list1 == null)
                _____________________;
        else
                _____________________;

        return newhead;
}
```

Answer:

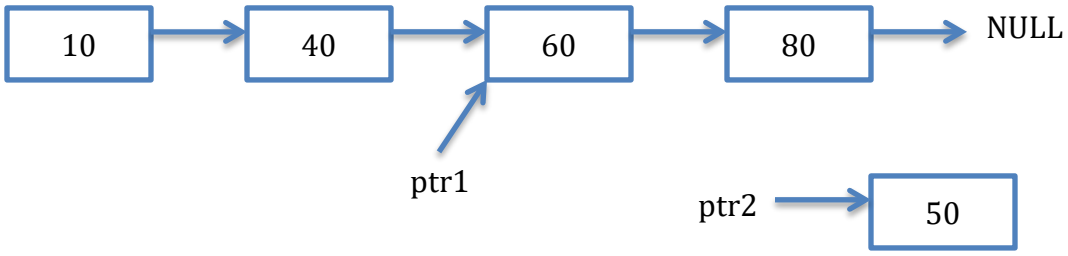
```
Node * Merge (Node * list1, Node * list2) {
        if (list1 == null) return list2;
        if (list2 == null) return list1;

        Node * newhead,temp;
        if (list1->data < list2->data) {
                newhead = list1;
                list1= list1->next;
        }
        else {
                newhead = list2;
                list2 = list2->next;
        }
        temp = newhead;

        while(list1!= null && list2 != null) {
                if (list1->data > list2->data){
                        temp->next = list2;
                        list2 = list2->next;
                }
                else {
                        temp->next = list1;
                        list1 = list1->next;
                }
                temp = temp->next;
        }
        if (list1 == null)
                temp->next = list2;
        else
                temp->next = list1;

        return newhead;
}
```

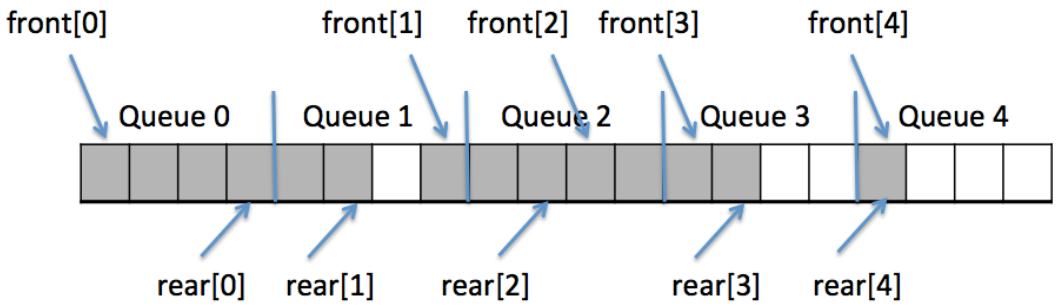
3. Consider the sorted linked list with these nodes:



The only pointer that is given for this list is ptr1 that is pointing to the node with the value 60 (shown above) and there is no head pointer provided. There is also a new node with the value 50 and a pointer, ptr2, pointing to it.
Correctly insert the new node to the above list in the sorted order. Explain your answer in sentences. No coding is required.

4. Postfix is preferable than infix for computers because of having no checks for parenthesis and no check for operator precedence rules. Prefix has also the same properties as postfix for having no parenthesis and precedence check. Can you explain why postfix is more preferable than prefix?

5. Multiple stacks/queues have a recovery mode if it is full and needs to add more items. The main idea is finding another stack/queue that has a space, which we can use by shifting the items. A sample multi queue is given below where there are 5 queues and in each of them there are 4 positions. Queue 0 and Queue 2 are full and others are not. For each queue a front and a rear pointer is kept.



Explain the steps that need to be taken when adding to a full queue and take recovery steps. Use short sentences. No coding is required.

Burada bu kadar birakabiliriz veya

Explain the steps that need to be taken when adding to a full queue and take recovery steps. Write only steps needed to update for the full query and its items inside. Use short sentences. No coding is required. (Hint there are 2 scenarios)

Veya

Ayni sekilde ve altta asagidaki gibi bosluklar

Step 1. Check the queues in the right side to find an empty space
Step 1.1 If found move the queues
Step 1.2 move data inside the full queue
Step 1.2.1. _____ (fill here)

Step 2 If no empty queue on the right, check the queues in the left side to find an empty space
Step 2.1 If found move the queues
Step 2.2 move data inside the full queue
Step 2.2.1. _____ (fill here)