1	4	3	2	5	2

create another linked list of same size.
count elements which are less than 3, store 3 at the value of counter
check elements one by one. ~~which are less than 3~~ store before 3 .

National University of Computer and Emerging Sciences, Lahore Campus

Course:	Data Structure	Course Code	
Program:	BSCS	Semester	4th
		Section	4B
Name:	Iman Sohail	Time	20 mins
Registration #:		Assessment	Quiz 2

Q1: Given a linked list and a value x, partition it such that all nodes less than x come before nodes greater than or equal to x.

7

Example:

Input: head = 1->4->3->2->5->2 and x = 3

Output: 1->2->2->4->3->5

a. Write down an algorithm for the aforementioned problem (Use any combination of your own imagination)

b. Write down code in C++ to accomplish the aforementioned task.

1	4	3	2	5	2

```
node * fun (int x){
node * p = head
int count = 0;
while (p! =nullptr){
    p = p -> next;
    count++; }
List obj;
while (i != counter)
obj.insert (0);
p = head; ~~int i=0;~~
~~while (i =~~
while (p! =nullptr)
    if (p-> data < 3)
obj. insert ( p -> data);
    p = p -> next;
~~obj.insert (x);~~ p = head;
while (p ! = nullptr)
    if (p->data > 3)|| p-> data==3)
    { obj. insert ( p-> data);
    p = p-> next;
}
return obj.head;
```

// creating a new linked list

// inserting values less than 3 by calling insert at end function

// inserting values greater/equal to 3 after the value

// returning the newly created list

```
void insert ~~at end~~ (int x)
{
if (head . . nullptr)
{
head->data = x;
head->next = nullptr;
}
else
{ ~~node~~ node * p = head;
while ( p! = nullptr);
    p = p->next;
}
p -> data = x;
p -> next = nullptr
}
```

National University of Computer and Emerging Sciences, Lahore Campus

	Course:	Data Structure	Course Code:	
	Program:	BSCS	Semester:	4th
			Section:	4B
	Name:	Mukaram Murtaza	Time:	20 mins
	Registration #:	L17-4116	Assessment	Quiz 2

Q1: Given a linked list and a value x, partition it such that all nodes less than x come before nodes greater than or equal to x.

5

Example:

Input: head = 1->4->3->2->5->2 and x = 3

Output: 1->2->2->4->3->5

a. Write down an algorithm for the aforementioned problem (Use any combination of your own imagination)

b. Write down code in C++ to accomplish the aforementioned task

a) ~~if(head!=N~~

Creating 2 linked lists, Lower & Greater

~~if check~~

Traversing through the original Linked List and storing the element in Lower if it is lesser than 3. & storing the element in greater if it greater than 3. Finally merging the 2 Linked Lists.

b) Code:

```
void Partition(LinkedList LL, key)
{ if(head == NULL)
    { return ;
else { LinkedList lower, greater;
       aux = head;
while (aux -> next! = NULL)
    { if(aux -> data < key)
        { lower.insertAtStart(aux->data)
```

(continued on next page)

```
else{
    greater.insertAtStart (aux->data);
}
aux = aux->next;
}
}  ~~lower.~~
while (lower.aux->next != NULL)
{ lower.aux = lower.aux->next;
}
lower.InsertAtEnd (key); // stores the key(3) in the end of ~~first linked list.~~
lower.Merge (greater);  // Merges the two linked list.
                        The greater linked list is
                        linked to end of lower linked list.
```

National University of Computer and Emerging Sciences, Lahore Campus

Course:	Data Structure	Course Code:	
Program:	BSCS	Semester:	4th
		Section:	4B
Name:	Umar Iqbal	Time:	20 mins
Registration #:	18-1156	Assessment	Quiz 2

Q1: Given a linked list and a value x, partition it such that all nodes less than x come before nodes greater than or equal to x.

Example:

Input: head = 1->4->3->2->5->2 and x = 3

Output: 1->2->2->4->3->5

2

a. Write down an algorithm for the aforementioned problem (Use any combination of your own imagination)

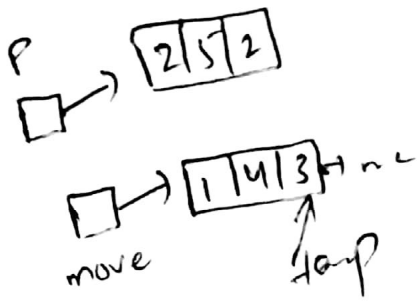
b. Write down code in C++ to accomplish the aforementioned task

(a) (i) Divide linked list into two parts after value x. and both points to nullptr

ii) Check value which is greater or equal than x will insert at the end of 1st list after x.

iii) Insert the remaining nodes of the 2nd list to the 1st list.

```
void arrange (List & L, T x)
{
    Node<T> *temp = L.head;
    Node<T> * p = nullptr;
    Node<T> * move = temp;
while (temp != nullptr && temp->data != x)
        temp = temp->next;
    move = temp->next
p = temp->next
temp->next = nullptr;
```



```
while (p->next != nullptr)
{
    if (p->data > x)
    {
        temp->next = p;
        temp = temp->next;
        temp->next = nullptr;
    }
    else what?
}
Node <T> *r = nullptr
while (move->data != x)
{
    ~~if (move->data~~
    if ~~while~~ (temp->next != nullptr)
    {
        ~~move~~
        ~~p = temp~~;
        move->next = temp;
        temp = temp->next;
    }
}
```

Insert at biginning