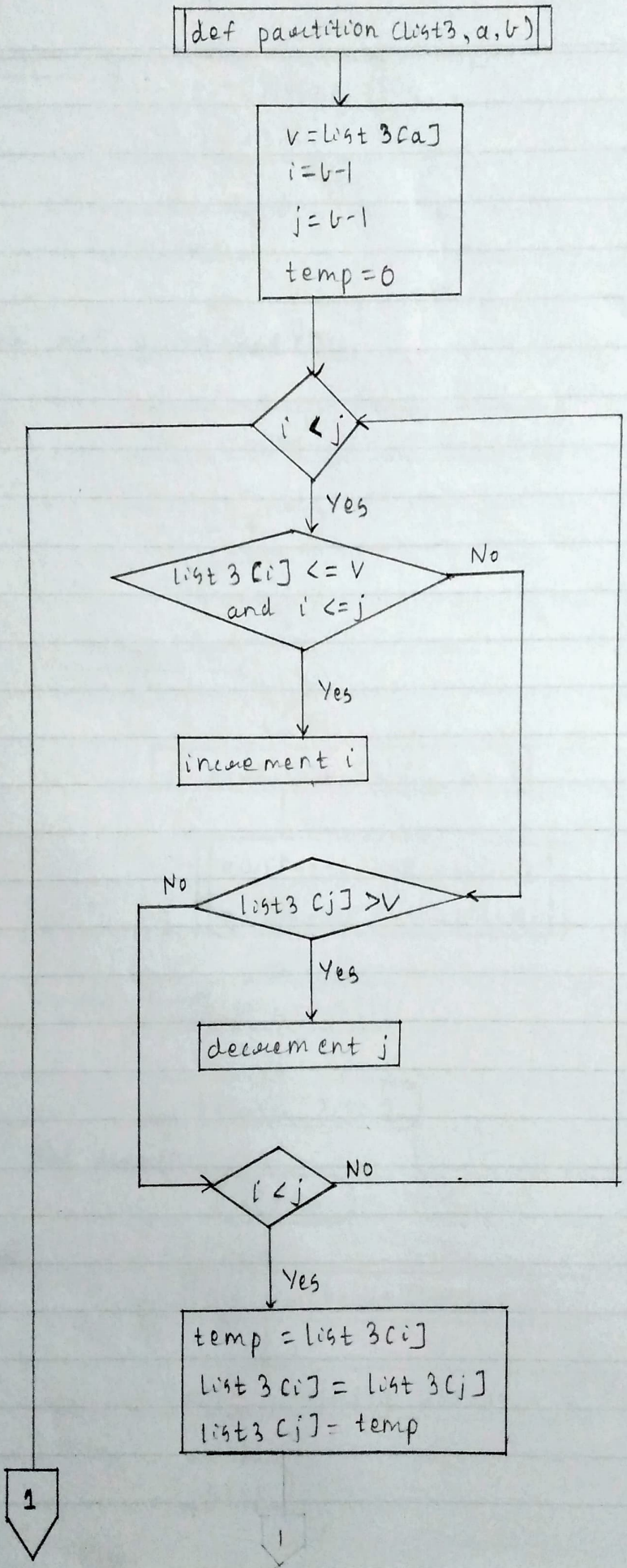
→ Lists:

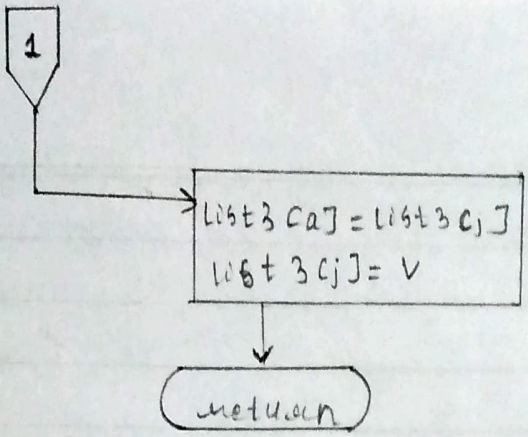
- List is a sequence data type which is used to store the collection of data.
- The elements of lists are separated by commas (,) and enclosed within square brackets '[]'.
- Lists are mutable, meaning the elements are interchangeable after it has been created.
- Since lists are indexed, lists can have items with the same value.
- eg:

```
# Declaration of list
list1 = [2, "machkar", "orange"]
print("The list is: ", list1)
list1[1] = "Tanmay" # changing element of list1
print("List after element is changed: ", list1)
```

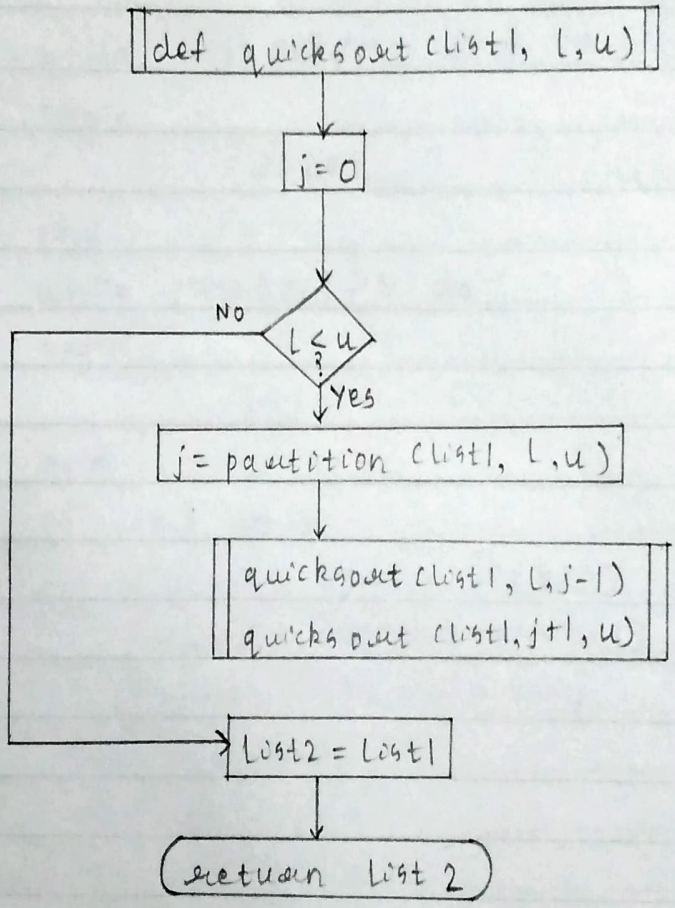
```
# Output:
The list is: [2, 'machkar', 'orange']
List after element is changed: [2, 'Tanmay', 'orange']
```

owchart for def partition

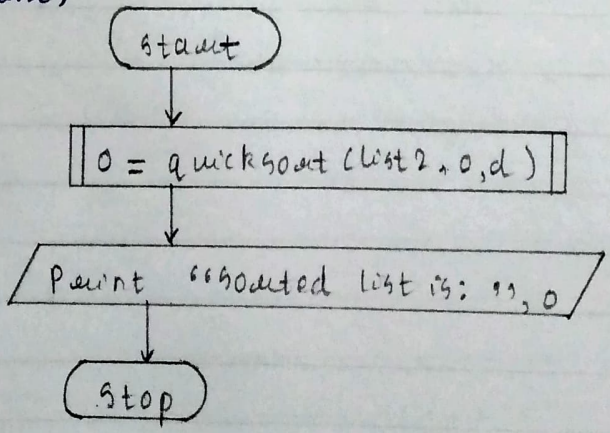




→ Flowchart for def quicksort()



→ Flowchart for def main()



→ Pseudocode for def Partition()

1. Initialize v = list3 [a]
   Initialize i = a
   Initialize j = b-1
   Initialize temp = 0
2. while i<j do
   begin
      while list3 [i] <= v and i<=j do
      begin
         i = i+1
      end
      while list3 [j] > v do
      begin
         j = j-1
      end
      if i<j then
         store temp = list3 [i]
         store list3 [i] = list3 [j]
         store list3 [j] = temp
3. store list3 [a] = list3 [j]
   store list3 [j] = v
4. return j

→ Pseudocode for
def quicksort()

1. Initialize j = 0
2. if l<u then
   store j = partition (list1, l, u)
   call function quicksort (list1, l, j-1)
   call function quicksort (list1, j+1, u)
3. store list2 = list1
4. return list2

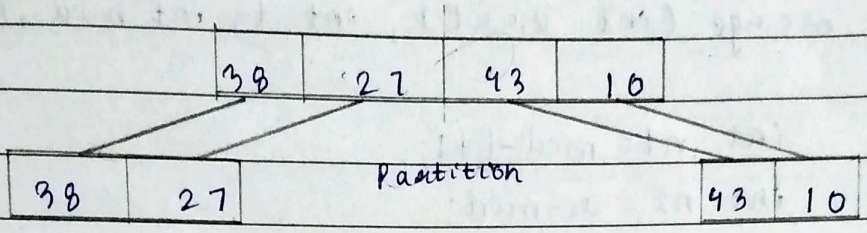
→ Pseudocode for def main()

1. ~~Initialize o = quicksort (list2, 0, d)~~ Start
2. Initialize o = quicksort (list2, 0, d)
3. Display sorted list is, o
4. Stop

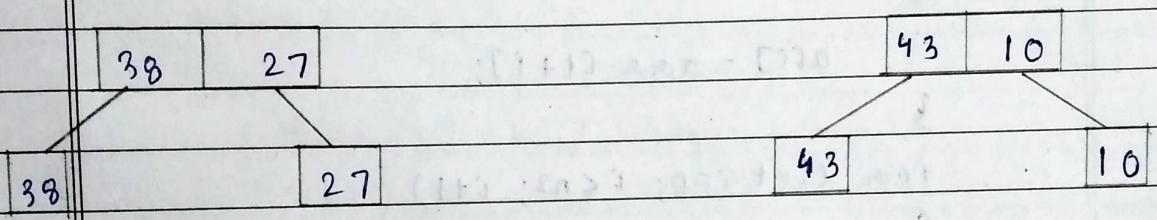
Q1. Explain merge sort with example and write C++ program for same.

Ans. Let us consider an array arr[] = {38, 27, 43, 10}

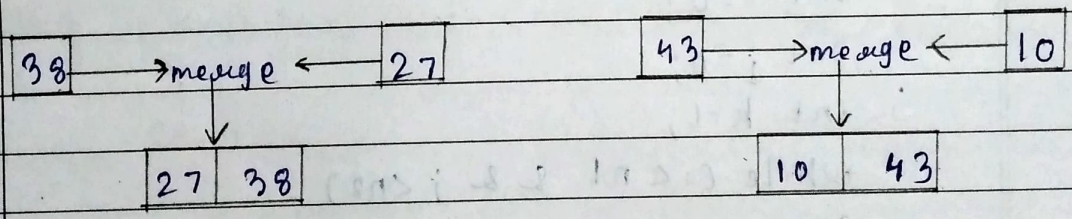
Step1: Initially divide the array into two equal halves:



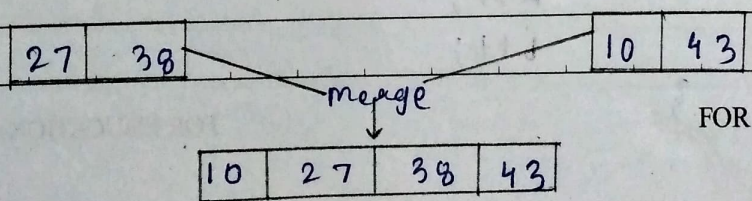
Step2 These subarrays are further divided into two halves. Now they become array of unit length that can no longer be divided and array of unit length are always sorted.



Step 3: These sorted ~~arr~~ subarrays are merged together and we get bigger sorted subarrays.



Step 4: The merging process is continued until the sorted array is built from the smaller subarrays.



→ Program:

```
#include <iostream>
using namespace std;

void merge (int arr[], int l, int mid, int r)
{
    int n1 = mid-l+1;
    int n2 = r-mid;

    int a[n1];
    int b[n2];

    for (int i=0; i<n1; i++)
    {
        a[i] = arr[l+i];
    }
    for (int i=0; i<n2; i++)
    {
        b[i] = arr[mid+1+i];
    }
    int i=0;
    int j=0;
    int k=l;
    while (i<n1 && j<n2)
    {
        if(a[i] < b[j])
        {
            arr[k] = a[i];
            k++;
            i++;
        }
```

```
        else
        {
            arr[k] = b[j];
            k++;
            j++;
        }
    }
    while (i<n1)
    {
        arr[k] = a[i];
        k++;
        i++;
    }
    while (j<n2)
    {
        arr[k] = b[j];
        k++;
        j++;
    }
}
void merge sort (int arr[], int l, int r)
{
    if (l<r)
    {
        int mid = (l+r)/2;
        merge sort (arr, l, mid);
        merge sort (arr, mid+1, r);
        merge (arr, l, mid, r);
    }
}
```

```
int main()
{
    int arr[] = {38, 27, 43, 103};
    mergeSort(arr, 0, 3);
    for(int i=0; i<4; i++)
    {
        cout<<arr<<" ";
    }
    return 0;
}
// Output:
10    27    38    43
```