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

void Merger(int arr[],int c ,int v,int b){
int p,q,j,k;
q=v;
p=j=v-1;
cout<<q<<v<<p<<j<<endl;
while((c<=p)&&(q<=b)){
    if(arr[p]>arr[q]){
        p--;
        q++;
    }
    else if(arr[p]<arr[q]){
        break;
    }
}
k=q;
while(p<j){
    swap(arr[k],arr[j]);
    k--;
    j--;
}
for(int t=0; t<=b;t++){
    cout<<arr[t]<<" ";
}
cout<<endl;
}

void Breaker(int arr[],int L, int R){
      int m;
    if(R-L<2){
      m=L;
    }
    if (R-L>=2){
            m=(((R-L)+1)/2)+L;}
    cout<<m<<endl;
    if (R-L<2){
        if(arr[L]>arr[R]){
            cout<<arr[L]<<" "<<arr[R]<<endl;
            swap(arr[L],arr[R]);
        cout<<arr[L]<<" "<<arr[R]<<endl;
        }
    }
    else if(R-L>=2){
        Breaker(arr,L,m-1);
        Breaker(arr,m,R);
        for(int t=0; t<=R;t++){
    cout<<arr[t]<<" ";
}
cout<<endl;
    }
    Merger(arr,L,m,R);
}
void mergeSort(int arr[], int mySize){
    cout<<"Called"<<endl;
    int l=0;
    Breaker(arr,l,mySize-1);

}



int main()
{
```

The bottle neck is right here

```
    int arr[]={28,17,5,18,30,8};
    mergeSort(arr,6);
    for(int i:arr){
        cout<<i<<" ";
    }
    return 0;
}
```

Problem 2 Explanation:

First up the complexity. For one to be able to do mergesorting in place you need to first continue to split the array. Now when splitting you can simple keep calling the merge function just giving it different start and end bounds. That will in theory separate the array but it will all be part of the same array. Then once you do that you need to start piecing it back together. When putting it back together, you need to start from the middle and then find a region in which subArrayA elements is larger than subArrayB. Then you need to push the array. Example

A={15,29,32,64,91,}

B={11,19,27,80,89}

So the region would be

{15,29,32,64,91,11,19,27,80,89} In this 91>11,64>19, and 32>27. We stop there cause 29<80 so can't use that. Then we simple push till all the blue are in the left and yellow in the right

{15,29,27,32,64,91,11,19,80,89}->{15,29, 19,27,32,64,91,11 ,80,89}->{15,29, 11,19,27,32,64,91, 80,89}. Then we break that into smaller groups and keep going so that A and B are joined and sorted. The only problem I ran into is that visualizing the recursion is difficult and my code did the swap that I needed but when it can to printing/the final list, it wasn't sorting properly. One element in each block was just out of place. Also because of this I would not consider this sort to be stable. The method that the data is inputted in matter and I feel as if the slightest wrong move can cause an unexpected error somewhere in the program