Problem2:

Yes it is, because it has no place to fail. To ensure it is stable we must make sure that the Left half and Right half can either be greater than or equal too because that eliminates the possibility of error.

```
if (L[i] <= R[j])
```

The merge function takes a ⊖(n) time, also we are calling the merge function log (base 2) (n) times because we are splitting the initial array into two sizes hence the log (base 2) and we're recursively calling both half sized arrays therefore the running time would T(n) = 2T(n/2) + ⊖(n). Since we are recursively calling merge sort on A[0….n/2] and A[n/2+1….n] after the first iteration, then the bottleneck at A[n/2] after every recursive call. This will be true for both the best and worst cases.

```
void mergeSort(int arr[], int r)
{
    int m = r/2;


    mergeSort(arr, m);
    mergeSort(arr, m);

    merge(arr, 0, m, r);
}
```