Assignment 3

1) B: $4n^2 + 5n + 2$

C: Barometer operations; while(j<n), cout << arr[i] << arr[j], j++, cout << " ";

D: $O(n^2)$

2) B: $3n^2 + 13n + 3$

C: Barometer operations; while(j<=i), cout << j, j++

D: $O(n^2)$

3) B: Best case; 10n

This only happens when the entire array is filled with one number, for ex. [1,1,1,1,1]. When this happens, the nested while loop at most runs once, meaning it runs linearly.

Worst case; $(3n^2)/2 + (11n)/2 + 3$

This only happens when the array has no duplicates, for ex. [1,2,3,4,5]. This means that the nested while loop will slowly run more and more as the index increases, meaning it must run at quadratic time (nested loop).

C: Best case; everything inside the first while loop (int iResult, if(!duplicate), etc.)

Worst case; everything inside nested while loop (while condition, if(equal), iResult++)

D: Best case; $O(n)$ Worst case; $O(n^2)$

4) B: $3n^3 + 6n^2 + 4n + 4$

C: Barometer operations: while(iNext<rows), next+=m*m, iNext++ (in most inner loop)

D: $O(n^3)$

5) B: Best case; $(3n^2)/2 + (11n)/2 - 6$

C: Best case; everything in while loop **EXCEPT** for smallest=next, i.e., while(), if(), next++

This happens when the array is already sorted, so no need to keep changing the "smallest" index to a different one since it is already the smallest.

Worst case; everything in while loop, i.e., while(), if(), smallest=next, next++

This happens most of the time when using selection sort. Due to the fact that no matter what happens, the algorithm will always walk through the array n-1 times. So, while it is worse than the best case, it isn't much different since you are only doing one extra operation in the while loop; changing the "smallest" index a certain number of times.

D: Selection sort has the same best and worst order, therefore both are $O(n^2)$

6) B: $3n\log_2(n) + 17n - 6$

C: Barometer operations; everything in while loop i.e., while(ast<n), cout << "*", ast++

D: $O(n\log_2 n)$