CS330 Assignment 1

Question 1:

Gale and Shapley published their paper on the Stable Matching Problem in 1962; but a version of their algorithm had already been in use for ten years by the National Resident Matching Program, for the problem of assigning medical residents to hospitals.

Basically, the situation was the following. There were m hospitals, each with a certain number of available positions for hiring residents. There were n medical students graduating in a given year, each interested in joining one of the hospitals. Each hospital had a ranking of the students in order of preference, and each student had a ranking of the hospitals in order of preference. We will assume that there were more students graduating than there were slots available in the m hospitals.

The interest, naturally, was in finding a way of assigning each student to at most one hospital, in such a way that all available positions in all hospitals were filled. (Since we are assuming a surplus of students, there would be some students who do not get assigned to any hospital.)

We say that an assignment of students to hospitals is **stable** if neither of the following situations arises.

First type of instability: There are students s and s', and a hospital h, so that:

- s is assigned to h;
- s' is assigned to no hospital;
- h prefers s' to s.

Second type of instability: There are students s and s', and hospitals h and h', so that:

1) s is assigned to h;
2) s' is assigned to h' hospital;
3) h prefers s' to s;
4) s' prefers h to h'.

There is an instance: 5 students $S = \{1, 2, 3, 4, 5\}$ and two hospitals $H = \{C, M\}$, each with 2 positions available. The preference is given below. Give a stable matching and an unstable matching.

$C: < 5, 1, 2, 4, 3 >$; $M: < 5, 3, 1, 2, 4 >$;

$1: < C, M >$; $2: < C, M >$; $3: < C, M >$; $4: < M, C >$; $5: < M, C >$.

Implement the above algorithm using C or C++ with the provision to read an input preference text file as shown above (i.e. preference list). Output a list of stable matches. Validate your implementation with sufficient test cases.

Question 2:

Consider the following algorithm.

ALGORITHM Secret($A[0], A[1], \cdots, A[n-1]$)
// Input: an array $A[0], A[1], \cdots, A[n-1]$) of n numbers
// Output: an n by n array $\{B[i][j] \mid 0 \leq i, j \leq n-1\}$

1. **for** $i \leftarrow 0$ **to** $n-1$ **do**
2. **for** $j \leftarrow i$ **to** $n-1$ **do**
3. Add up array entries $A[i]$ through $A[j]$
4. Store the result in $B[i][j]$
5. **return** B

a) Suppose input $A = \{1, 2, 3, 4\}$, and each entry of B is initialized as 0, what is its output?

b) What is its basic operation?

c) How many times is the basic operation executed?

d) What is the efficiency class (Big O) of this algorithm?

Question 3:

Find the computational complexity of the following piece of code assuming that n=2m. Clearly show the detail working how complexity value is achieved.

```
for (int i = n; i > 0; i-- ) {
  for (int j = 1; j < n; j *=2) {
    for (int k = 0; k < j; k++) {
      // constant number of operations
    }
  }
}
```