Question1

"Explain how the program `code.c` works in detail and describe the worst time complexity for each function in the program."

Name: Anastasia Marinakou

AM: 1115202400120

`code.c` implements a `bubblesort` function and a `main` function , which the latter acts as the interface for the use (via command line).

Furthermore, the program defines a few macros to compare (`#define less(A, B) (key(A) < key(B))`), swap (`#define exch(A, B) { Item t = A; A = B; B = t; }`) and a combination of both: to swap two elements only if the first one is larger than the second (`#define compexch(A, B) if (less(B, A)) exch(A, B)`).

- `bubble()`

`bubble()` will take three parameters: 1. an array `a[]` of type `Item` (`int`), 2. a starting index (`l`) of type `int` , 3. an ending index (`r`) of type `int` as well, and will perform a typical bubble sort:

For all elements in the array (`for (i = l; i < r; i++)`), iterate from the last element to the `i` -th element (`for (j = r; j > i; j--)`) and swap all the `i` , `j` elements that are in inverse order, using the macro `compexch(a[j-1], a[j])` .

By the time `i` gets to `r` and the `for` loop ends, the array will be sorted.

Time complexity: For the first `for` loop the complexity will be O(N) and for the nested one will also be O(N) (*considering worst case*).

So the final **worst case time complexity** will be: O(N) * O(N) = O(N^2)

- `main()`

`main` will take two arguments from the command line: the first one is the amount of numbers that the user wants to sort (`N`) and the second one, describes if the user wants to sort random numbers, or to import them manually (`sw`).

Then, `main` will allocate space for the array that will contain the numbers (`a[N]`).

If `sw == 1` (randomized input), the program will use function `rand()` to assign to the `i` -th element of array `a[]` a **floating point number in range [0, 1000]**.

If the user chooses to import the numbers manually (`sw == 0`), the program will scan the numbers from `stdin` and assign them to each element of `a[]` .

`main` will then call `bubble` to sort `a[]` , by giving as parameters the array (`a`), the starting index (`0`) and the ending index (`N-1`).

Finally, the program will print to `stdout` the sorted array.

It must be noted that the program does not correctly handle the case where the user provides an incorrect amount of arguments, which could lead to **undefined behavior** or a **segmentation fault**. Moreover, main does not free the allocated space for (`a[]`), which could lead to memory leaks.

- Time complexity (`rand()` is considered to be O(1)):
  - O(N), for either :

  ```
  for (i = 0; i < N; i++)
  a[i] = 1000*(1.0*rand()/RAND_MAX);
  ```

  or:

  ```
  while (scanf("%d", &a[N]) == 1) N++;
  ```

  +

  - O(N), to print the final array `a[]` :

  ```
  for (i = 0; i < N; i++) printf("%3d ", a[i]);
  ```

  +

  - O(N^2), from `bubble()` :

```
void bubble(Item a[], int l, int r)
  { int i, j;
    for (i = l; i < r; i++)
      for (j = r; j > i; j--)
        compexch(a[j-1], a[j]);
  }
```

So the worst case time computational complexity would be:

O(N) + O(N) + O(N^2) = O(N^2)

Where we are only keeping O(N^2) because it's the **dominant** complexity class.