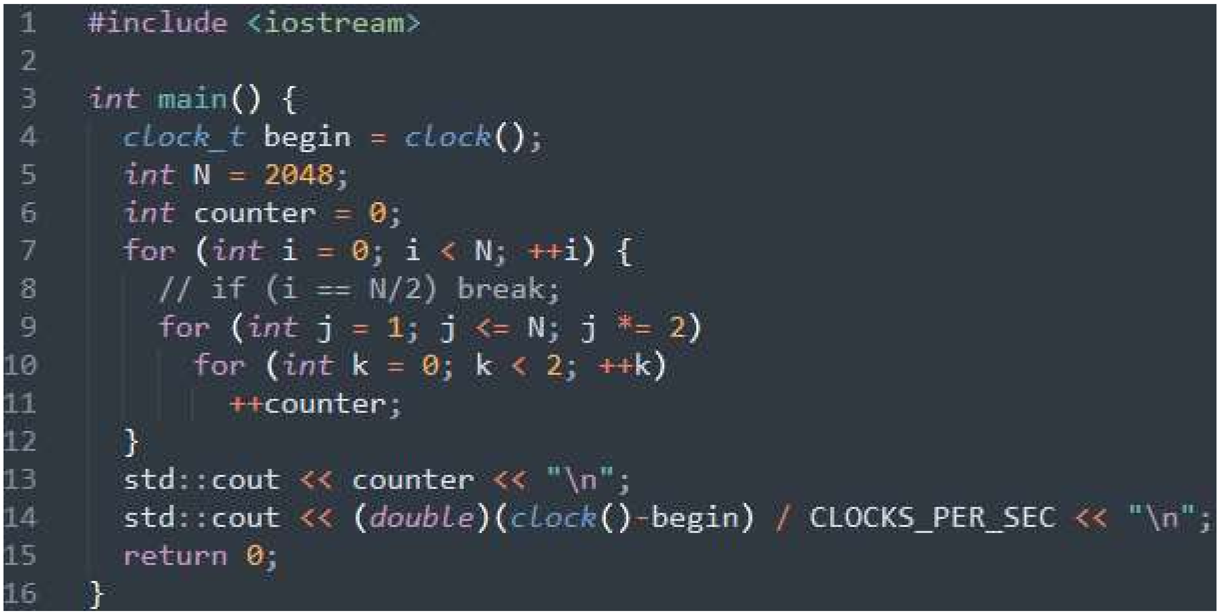
1. Fill in the blanks

(15 marks)

The following snippet of C++17 code has 3 nested loops...



```
 1  #include <iostream>
 2
 3  int main() {
 4    clock_t begin = clock();
 5    int N = 2048;
 6    int counter = 0;
 7    for (int i = 0; i < N; ++i) {
 8      // if (i == N/2) break;
 9      for (int j = 1; j <= N; j *= 2)
10        for (int k = 0; k < 2; ++k)
11          ++counter;
12    }
13    std::cout << counter << "\n";
14    std::cout << (double)(clock()-begin) / CLOCKS_PER_SEC << "\n";
15    return 0;
16  }
```

If you understand what we discuss in class, answer the following questions:

PS: Note that the valid options for time complexity analysis are: "1|log N|N|N log N|N^2|N^2 log N|N^3|2^N|N!".

1. What will be printed if we run that code? (any answer with the **first two most significant digits** correct will be accepted) 1 49152

2. What is the time complexity of the code above in terms of N? O(2). N log N

3. If N in line 5 is changed to 1000000 (1 Million), do you think your code can run in less than 1s if your computer can do 100 Million basic operations in 1s? Write Y/N in the box: 3 Yes

4. If the comment "//" in line 8 is removed so that line 8 is part of the code, will the time complexity that you have answered in part 2 changes? Write Y/N in the box: 4

5. If line 10 is changed to for (int k = 0; k < N; ++k), the time complexity of the code above in terms of N changes into O(5)

Enter the correct answer below.

1

Please enter a number for this text box.

Character Limit: 5

2

Character Limit: 10

3 □
Character Limit: 1

4 □
Character Limit: 1

5 □
Character Limit: 10

2. Fill in the blanks

(15 marks)

The following C++17 code is supposed to do these:

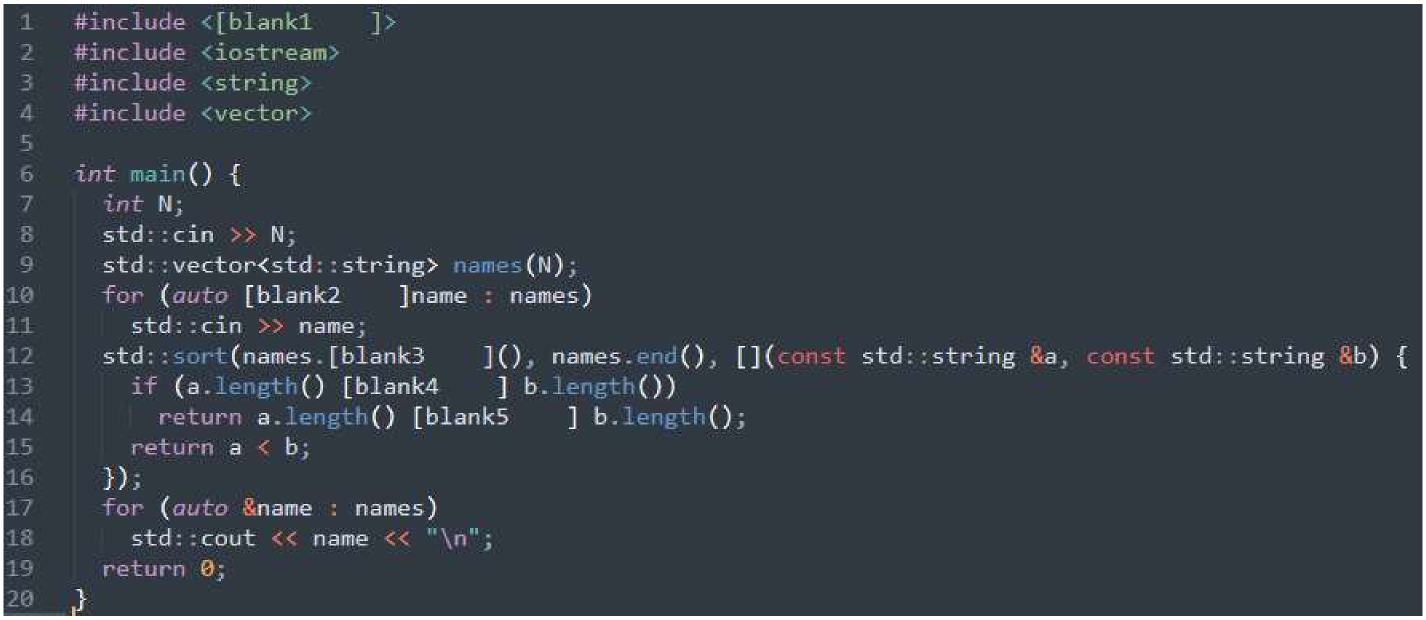
-. Read in an integer N in the first line and N distinct strings (of only lowercase alphabet ['a'..'z']) in the next N lines.

-. Store these N strings in a C++ std::vector

-. Sort these N strings based on non-decreasing lengths, e.g., shorter string first, and if ties, by lexicographic order.

-. Print the sorted N strings in N lines.

Unfortunately, there are 5 small subtle bugs in the code below. Fix them. Note that the grading is super strict and there is only one possible 'one-word' answer to each blank.



```
#include <[blank1    ]>
#include <iostream>
#include <string>
#include <vector>

int main() {
  int N;
  std::cin >> N;
  std::vector<std::string> names(N);
  for (auto [blank2    ]name : names)
    std::cin >> name;
  std::sort(names.[blank3    ](), names.end(), [](const std::string &a, const std::string &b) {
    if (a.length() [blank4    ] b.length())
      return a.length() [blank5    ] b.length();
    return a < b;
  });
  for (auto &name : names)
    std::cout << name << "\n";
  return 0;
}
```

```
1. #include <1>
```

```
2. for (auto 2name : names)
```

```
3. std::sort(names.3(), names.end(), [](const std::string &a,
const std::string &b) {
```

```
4. if (a.length() 4 b.length())

5. return a.length() 5 b.length();
```

Enter the correct answer below.

1

Character Limit: 10

2

Character Limit: 5

3

Character Limit: 7

4

Character Limit: 5

5

Character Limit: 5

Another New ADT

This is the harder part of the quiz. All the best.

3. Suppose you are tasked to design a**nother** ‘New’ Abstract Data Type (ADT) that needs to support the following three crucial operations.

1. add(v) - add item/data v into your ADT (you can assume that v is a non-negative integer),

2. count() - return the number of items currently in your ADT,

3. returnAnyWithEqualProbability() - return any item (any integer) that is currently in your ADT, or -1 (to signal that there is nothing currently in your ADT), and subsequently remove that item from y our ADT. The meaning of "Equal Probability" is as follows: If your ADT currently contains **N** items, each item has exactly 1/**N** chance to be the one returned (and then removed) by this function.

Example: Let's say **N** = 4 and we currently have 4 integers inside our ADT {7, 10, 11 and 21}. At this point, count() should return 4. Each of the integer is equally likely (1/4 or 25% chance each) to be returned (and then removed) by returnAnyWithEqualProbability(). Suppose, the chosen integer is 7, our ADT will then contains **N** = 3 integers {10, 11 and 21}.

Please select your best data structure to implement this another ‘New’ ADT and its three crucial operations. You can use any data structure that has been discussed in CS2040C (or beyond). (20% of the marks)

Please describe how you will actually implement these three crucial operations and analyze their time complexities! (10%/10%/60% of the marks for add(v), count(), and returnAnyWithEqualProbability(), respectively).

You will be graded based on the efficiency of your implementation (full 20 marks for the best answers)

(20 marks)

List Questions

We want to insert **N** integers of array **A** into the **middle position** of a **certain data structure** one by one. Note that if **k** is the size of the data structure before the push, the insertion index is **(k+1)/2** (using 0-based indexing).

For example, if **N** = 4, and **A** = {3, 9, 5, 1}, then the data structure will look like this after each insertion step:

-3 (index 0) after inserting 3 at index (0+1)/2 = 0
-3 (0) -> 9 (index 1) after inserting 9 at index (1+1)/2 = 1
-3 (0) -> 5 (index 1) -> 9 (2) after inserting 5 at index (2+1)/2 = 1
-3 (0) -> 5 (1) -> 1 (index 2) -> 9 (3) after inserting 1 at index (3+1)/2 = 2

Now answer the following sub-questions.

4.

The data structure is a **DoublyLinkedList DLL** with **both head and tail pointers.**

This DLL can choose to visit index **(k+1)/2** either from the head and advancing as many steps as necessary or from the tail and going backwards as many steps as necessary.

What is the overall time complexity needed to insert all **N** integers with this chosen data structure?

(4 marks)

○

O(n^2)

○

O(1)

○

O(n)

○

O(log n)

○

O(n log n)

○

Wait, this pseudo-code is actually incorrect...

5.

The data structure are **two Stacks: S1 and S2**.

We insert the next integer into S1, S2, S1, S2, ..., and so on in alternating fashion.

At the end we dump all contents of S1 into S2 (in reversed order) and finally store all contents of S2 (in reversed order) in array A again.

What is the overall time complexity needed to insert all **N** integers with this chosen data structure?

(4 marks)

○

O(n log n)

○

O(1)

○

O(log n)

○

O(n)

○

O(n^2)

○

Wait, this pseudo-code is actually incorrect...

6.

The data structure is a **Min Priority Queue (likely Binary Min Heap).**

We insert pair of **((k+1)/2, next-value-of-A)** into the Min Priority Queue one by one.

At the end extractMin() **N** times to get the desired ordering (the second field of the pair).

What is the overall time complexity needed to insert all **N** integers with this chosen data structure?

(4 marks)

○

O(1)

○

O(log n)

○

O(n log n)

○

O(n^2)

○

Wait, this pseudo-code is actually incorrect...

○

O(n)

7.

The data structure is a **SingleLinkedList SLL** with **only the head pointer.**

This SLL has no other way to visit index **(k+1)/2** other than starting from the head and advancing as many steps as necessary.

What is the overall time complexity needed to insert all **N** integers with this chosen data structure?

(4 marks)

○

O(1)

○

O(n log n)

○

O(n^2)

○

O(log n)

○

O(n)

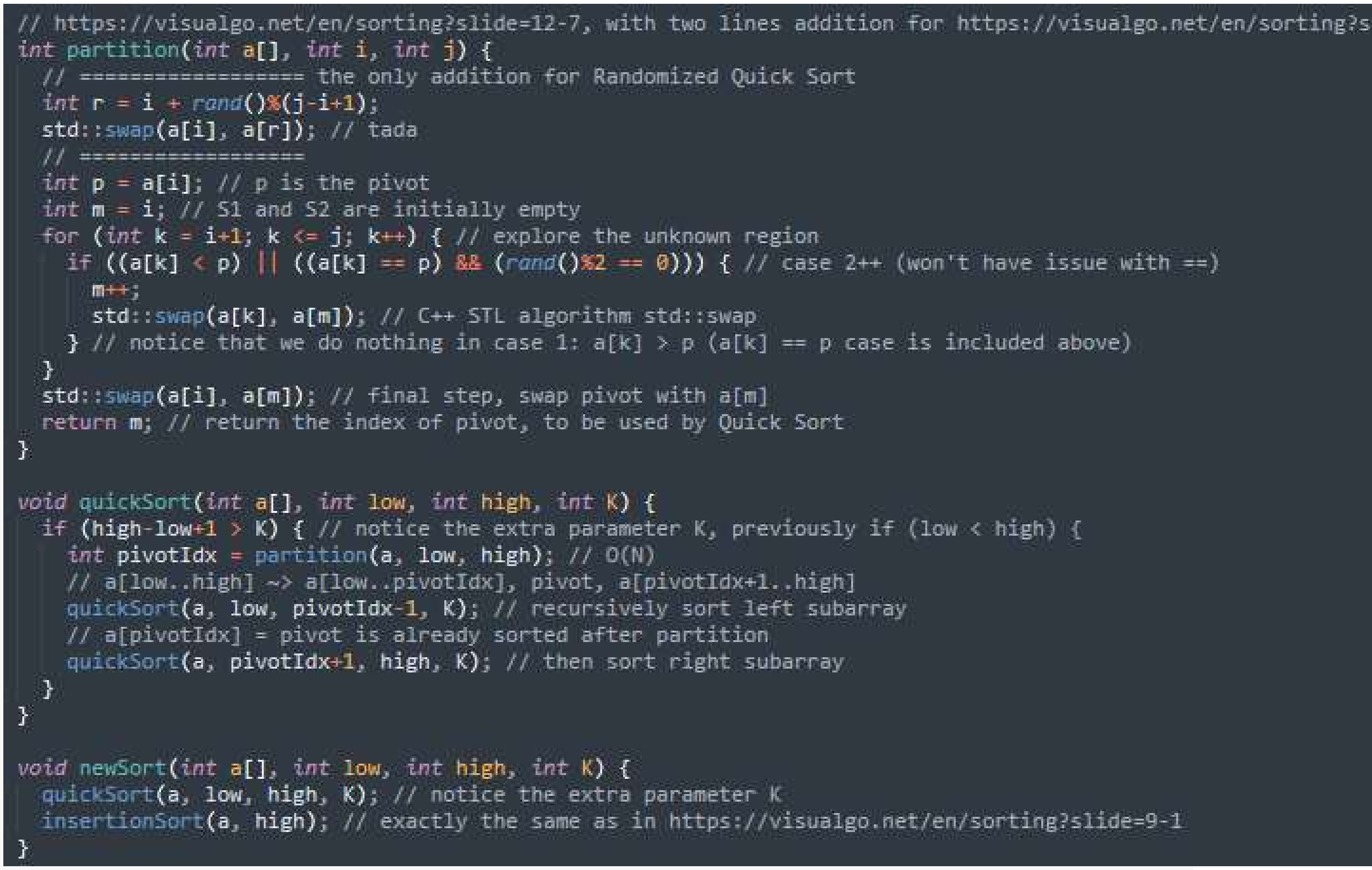
○

Wait, this pseudo-code is actually incorrect...

A New Sorting Algorithm

A "new" sorting algorithm...

Let's define a "new sorting algorithm" newSort() as follows:



```
// https://visualgo.net/en/sorting?slide=12-7, with two lines addition for https://visualgo.net/en/sorting?s
int partition(int a[], int i, int j) {
  // ================== the only addition for Randomized Quick Sort
  int r = i + rand()%(j-i+1);
  std::swap(a[i], a[r]); // tada
  // ==================
  int p = a[i]; // p is the pivot
  int m = i; // S1 and S2 are initially empty
  for (int k = i+1; k <= j; k++) { // explore the unknown region
    if ((a[k] < p) || ((a[k] == p) && (rand()%2 == 0))) { // case 2++ (won't have issue with ==)
      m++;
      std::swap(a[k], a[m]); // C++ STL algorithm std::swap
    } // notice that we do nothing in case 1: a[k] > p (a[k] == p case is included above)
  }
  std::swap(a[i], a[m]); // final step, swap pivot with a[m]
  return m; // return the index of pivot, to be used by Quick Sort
}

void quickSort(int a[], int low, int high, int K) {
  if (high-low+1 > K) { // notice the extra parameter K, previously if (low < high) {
    int pivotIdx = partition(a, low, high); // O(N)
    // a[low..high] ~> a[low..pivotIdx], pivot, a[pivotIdx+1..high]
    quickSort(a, low, pivotIdx-1, K); // recursively sort left subarray
    // a[pivotIdx] = pivot is already sorted after partition
    quickSort(a, pivotIdx+1, high, K); // then sort right subarray
  }
}

void newSort(int a[], int low, int high, int K) {
  quickSort(a, low, high, K); // notice the extra parameter K
  insertionSort(a, high); // exactly the same as in https://visualgo.net/en/sorting?slide=9-1
}
```

In pseudo-code:

1) newSort actually perform randomized quick sort (as discussed in VisuAlgo/class) starting from low = 0 and high = n-1 with a special parameter K. As long as the subarray length is bigger than K (e.g., high-low+1 > K). However, once high-low+1 <= K, we stop the randomized quick sort from recursing deeper and will return.

2) At the end, newSort performs insertion sort (as discussed in VisuAlgo/class) on the full array A from 0 to n-1.

8.

Write a short explanation of why newSort() remains a correct sorting algorithm?

(6 marks)

Character Limit: 500Word Limit:

9.

Fill in the blanks

(6 marks)

A). If the special parameter K is set to N, what is the time complexity of newSort()?

Your answer for A). is O(1)

B). If the special parameter K is set to 1, what is the time complexity of newSort()?

Your answer for B). is O(2)

Enter the correct answer below.

1

Character Limit: 7

2

10.

What is the expected time complexity of newSort() in terms of K (that can range from 1 to N) and N?

This probably requires manual grading. So, justify your analysis in the essay box.

(8 marks)

Character Limit: 500Word Limit:

One Sorting Application

Given an array **A** containing **N** integers that can contain duplicates, each integer is between [1..100K], sort the **N** integers so that given any two integers **U** and **V**, **U** appears before **V** if the frequency of appearance of **U** in **A** is *larger* than the frequency of **V** in **A**. If their frequency of appearances are equal, the integer which appears *earlier* in the **A** should also appear earlier in the sorted version of **A**. Print the first **K (1 <= K <= min(777, N))** integers of the sorted version of **A** at the end.

Example 1: **N** = **K** = 5, **A** = {2, 1, 2, 1, 2}, as 2 appears 3x and 1 appears only 2x, the sorted version of **A** should be = {2, 2, 2, 1, 1}.

Example 2: **N** = **K** = 9, **A** = {1, 3, 3, 3, 2, 2, 2, 1, 1}, we have three-way ties as 1, 3, 2, all appears 3x in **A**. However, the order of appearance is 1, 3, and then 2, so the sorted version of **A** should be = {1, 1, 1, 3, 3, 3, 2, 2, 2}.

Last Example 3: **N** = 9, **K** = 7, **A** = {1K, 3K, 1K, 99K, 5K, 1K, 2K, 2K, 3K} (notice 'large' positive integers not more than 100K?). If you have fully understood the problem, the first **K** = 7 integers in the sorted version of **A** should be = {1K, 1K, 1K, 3K, 3K, 2K, 2K}.

11.

Fill in the blanks

(5 marks)

If **N** = **K** = 7, **A** = {3, 4, 7, 2, 5, 1, 1}, the first integer in the sorted version of **A** is 1 and the third integer in the sorted version of **A** is 2. 1,1,3,4,7,2,5

If **N** = 8, **K** = 7, **A** = {9, 7, 9, 7, 3, 3, 3, 2}, the fourth integer in the sorted version of **A**, i.e., **A[3]** (after this sorting) is 3 and the second last integer in the sorted version of **A**, i.e., **A[6]** (after this sorting) is 4 3,3,3,9,9,7,7,2

If **N** = **K** = 5, **A** = {1, 5, 9, 2, 7}, will the sorted version of A be any different 5 (Y/N)? 1,5,9,2,7

Enter the correct answer below.

1
Please enter a number for this text box.
Character Limit: 2

2
Please enter a number for this text box.
Character Limit: 2

3
Please enter a number for this text box.
Character Limit: 2

4
Please enter a number for this text box.
Character Limit: 2

5 □
Character Limit: 1

12.

How are you going to solve this problem?

You can use any technique that we have learned in class and beyond. You are allowed to start from the skeleton template below. You can answer in pseudo-code or complete the entire C++ (17) code.

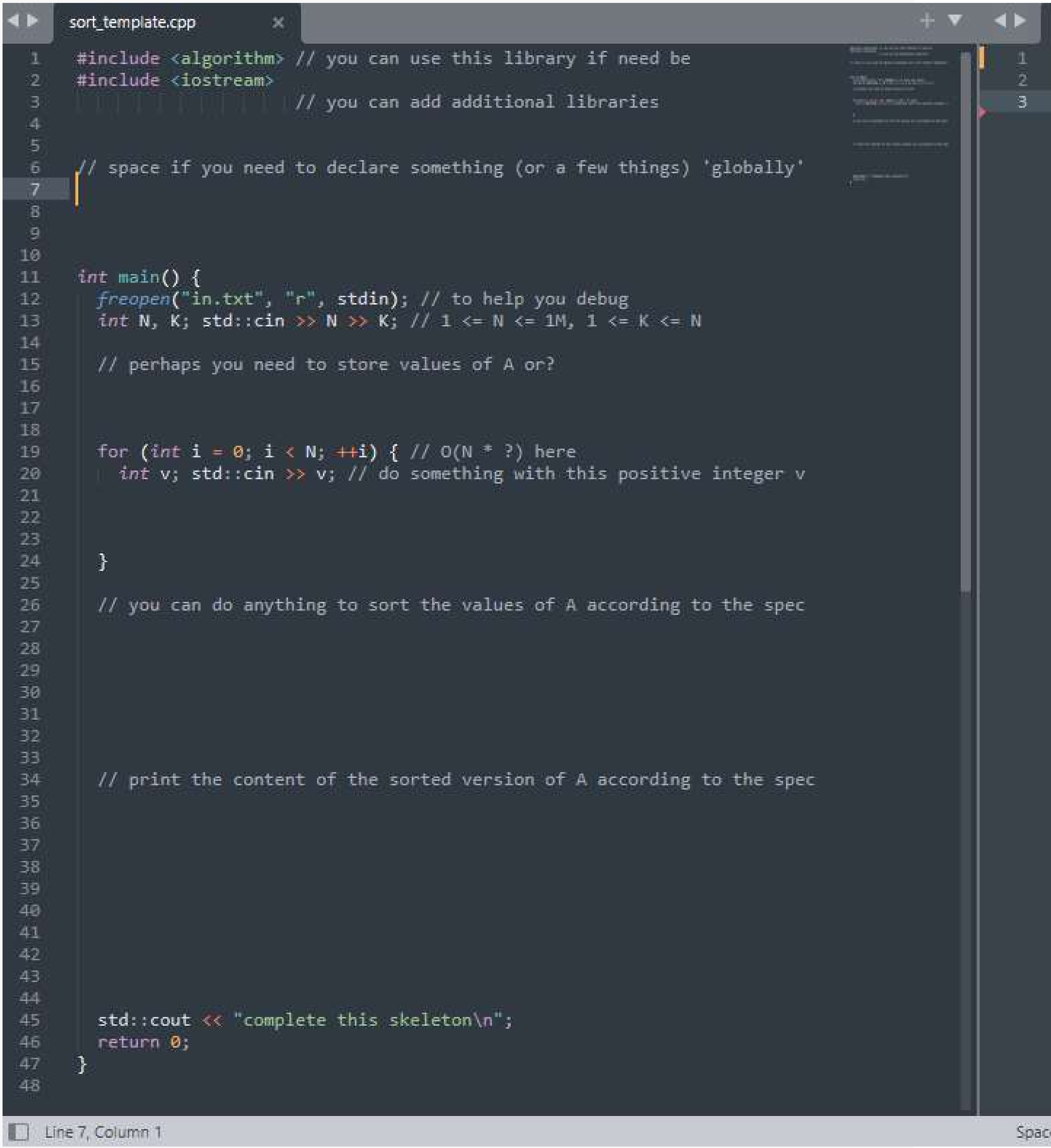
At the end, analyze the time complexity of your overall solution. You will be graded roughly as follows.

Full marks if your solution is the theoretical fastest for large **N** up to ~1M and **K** has this constraint (1 <= **K** <= **min(777, N)**) in 1s.

About half marks if your solution can only pass if **N** is limited to ~10K in 1s.

About one-fourth of the marks if your solution can only pass if all **N** integers **are distinct**.

Reference point: We assume that today's computer can do 10^8 or 100M operations in 1s.



```cpp
#include <algorithm> // you can use this library if need be
#include <iostream>
                     // you can add additional libraries


// space if you need to declare something (or a few things) 'globally'




int main() {
  freopen("in.txt", "r", stdin); // to help you debug
  int N, K; std::cin >> N >> K; // 1 <= N <= 1M, 1 <= K <= N

  // perhaps you need to store values of A or?



  for (int i = 0; i < N; ++i) { // O(N * ?) here
    int v; std::cin >> v; // do something with this positive integer v



  }

  // you can do anything to sort the values of A according to the spec







  // print the content of the sorted version of A according to the spec










  std::cout << "complete this skeleton\n";
  return 0;
}
```

(9 marks)

Character Limit: 3000Word Limit: