CSE 100
Lab (Programming) Assignments

Jan 17, 2023

Fall 2023

Sungjin Im

What are they?

- Your job is to implement some algorithms you learned in class.
  - There could be minor tweaks, but they are basically the same as the algorithms you will learn in class

- Examples:
  - Sorting algorithms
  - Minimum spanning tree algorithms
  - Algorithms based on dynamic programming
  - Divide-and-conquer algorithms
  - …

Why should we implement them?

- We will unlikely implement them after graduation and will probably use the codes others implanted. So, why?
  - They are so fundamental and you're expected to be able to implement them any time
    - It's no surprise that typical programming interviews ask you to show how to code them.
  - There are some subtleties you might have not realized when looking at the algorithm description. Implementing them helps you to understand them better.
  - Your major is Computer Science and Engineering: we want to not only understand how algorithms work but also demonstrate that they actually work.

Why don’t you ask us to implement more difficult algorithms?

- I’d love to. For example, I’d so love to give an assignment where you have to come up with an algorithm and implement it.

- But this course is a required course, meaning that I should focus on covering more fundamental algorithms.
  - I’m considering to create an elective course to cover ”Design and Analysis of Algorithms”, but it will be hard before we get more faculty lines

Example: Lab00

CSE100 Algorithm Design and Analysis — Lab 00, Fall 2021
Last update: 8/25/2021 — Deadline: See the Syllabus on CatCourses

---

Find Max and Min

This lab is indented to help you understand how you should test and submit your own code. Therefore, we provide a solution (yourid.cpp). This lab will be worth ZERO points but we still strongly encourage you to submit your code to see if you've completely understood how submission and grading work.

Description Given n numbers, we want to find the maximum and the minimum numbers.

Example: Lab00 (cont')

Input structure The input starts with an integer number which indicates the number of elements (integers) in the input sequence, n. Then, the elements in the sequence follow, one per line.

Output structure Output the maximum of all numbers in the sequence, followed by a semi-colon and the minimum number. There should be *no* white character in your output.

Example: Lab00 (cont’)

Examples of input and output:

Input

```
6
15
13
12
11
16
14
```

Output

```
16;11
```

See the lab guidelines for submission/grading, etc., which can be found in Files/Labs.

Where should I program?

- You can use any programming platforms you like.

- BUT, we will compile your code using the standard c++ 14 compiler.
  - That's another reason why we provide Makefile

- Your code must compile with the standard c++ 14 compiler and pass the text examples.
  - Using different compilers could result in you getting 0.

e.g. In my MAC

```
[sungjins-air-2:Labs sungjin$
[sungjins-air-2:Labs sungjin$
[sungjins-air-2:Labs sungjin$
[sungjins-air-2:Labs sungjin$
[sungjins-air-2:Labs sungjin$
[sungjins-air-2:Labs sungjin$
[sungjins-air-2:Labs sungjin$
[sungjins-air-2:Labs sungjin$
[sungjins-air-2:Labs sungjin$
[sungjins-air-2:Labs sungjin$
[sungjins-air-2:Labs sungjin$
[sungjins-air-2:Labs sungjin$
[sungjins-air-2:Labs sungjin$
[sungjins-air-2:Labs sungjin$
[sungjins-air-2:Labs sungjin$
[sungjins-air-2:Labs sungjin$
[sungjins-air-2:Labs sungjin$
[sungjins-air-2:Labs sungjin$ ls
 Lab00.zip
 sungjins-air-2:Labs sungjin$
```

```
sungjins-air-2:Labs sungjin$ unzip Lab00.zip
Archive:  Lab00.zip
  inflating: Grade
  inflating: Makefile
   creating: testfiles/
  inflating: testfiles/.DS_Store
  inflating: testfiles/o2
  inflating: testfiles/o5
  inflating: testfiles/o4
  inflating: testfiles/o3
  inflating: testfiles/t5
  inflating: testfiles/t2
  inflating: testfiles/t3
  inflating: testfiles/t4
  inflating: testfiles/o1
  inflating: testfiles/t1
  inflating: yourid.cpp
sungjins-air-2:Labs sungjin$ ls
Grade           Lab00.zip       Makefile        testfiles       yourid.cpp
sungjins-air-2:Labs sungjin$
```

```
[sungjins-air-2:Labs sungjin$ cd testfiles
[sungjins-air-2:testfiles sungjin$ ls
 o1      o2      o3      o4      o5      t1      t2      t3      t4      t5
[sungjins-air-2:testfiles sungjin$ cat t3
 3
 1
 2
 3
[sungjins-air-2:testfiles sungjin$ cat o3
[3;1sungjins-air-2:testfiles sungjin$
[sungjins-air-2:testfiles sungjin$ cat t4
 4
 1
 2
 3
 4
[sungjins-air-2:testfiles sungjin$ cat o4
 4;1sungjins-air-2:testfiles sungjin$
```

```
[sungjins-air-2:testfiles sungjin$ cd ..
[sungjins-air-2:Labs sungjin$ ls
 Grade            Lab00.zip       Makefile        testfiles       yourid.cpp
 sungjins-air-2:Labs sungjin$
```

```
#include <iostream>

#define MAX_INT 2147483647

using namespace std;


int main(int argc,char **argv) {

  int* Sequence;
  int arraySize = 1;

  // Get the size of the sequence
  cin >> arraySize;
  Sequence = new int[arraySize];

  // Read the sequence
  for(int i=0; i<arraySize; i++)
    cin >> Sequence[i];

  // Initializing the max and min numbers
  int max_num = Sequence[0];
  int min_num = Sequence[0];

  // Updating the max and min numbers
  for(int i=1; i<arraySize; i++)
  {
    if (Sequence[i] > max_num)
      max_num = Sequence[i];
    if (Sequence[i] < min_num)
      min_num = Sequence[i];
  }

  // output
  cout << max_num << ";" << min_num;


  // Free allocated space
  delete[] Sequence;

}
~
~
```

Lab00. We provided a solution, as this lab is to help you get familiar with how to test and submit your code

Your file name must be ”your email id.cpp”

```
sungjins-air-2:Labs sungjin$ cp yourid.cpp im3.cpp
sungjins-air-2:Labs sungjin$ g++ -std=c++14 -o a.exe im3.cpp
sungjins-air-2:Labs sungjin$
```

Edit Makefile accordingly

```
CC = g++

all:
	$(CC) -std=c++14 -o a.exe im3.cpp

clean:
	rm -f a.exe

~
```

This was yourid.cpp

You can test your code by entering an input by hands

```
sungjins-air-2:Labs sungjin$ ./a.exe
3
1
2
3
3;1sungjins-air-2:Labs sungjin$
```

You can test your code for each test example you're given

```
[sungjins-air-2:Labs sungjin$ ./a.exe < ./testfiles/t1
[1;1sungjins-air-2:Labs sungjin$ ./a.exe < ./testfiles/t2
[2;1sungjins-air-2:Labs sungjin$ ./a.exe < ./testfiles/t3
 3;1sungjins-air-2:Labs sungjin$
```

You can use the grading too you're given

```
sungjins-air-2:Labs sungjin$ ./Grade
   Correct for 1 th example.
   Correct for 2 th example.
   Correct for 3 th example.
   Correct for 4 th example.
   Correct for 5 th example.
sungjins-air-2:Labs sungjin$
```

Your execution file name must be a.exe

You may need to change the permision

```
sungjins-air-2:Labs sungjin$ chmod 700 Grade
sungjins-air-2:Labs sungjin$
```

To see the test result in detail:

```
sungjins-air-2:Labs sungjin$ cat result
==================
test 1:
student:
1;1
grader:
1;1
==================
test 2:
student:
2;1
grader:
2;1
==================
test 3:
student:
3;1
grader:
3;1
==================
test 4:
student:
4;1
grader:
```

Submit your code using catcourses

- Remember that your file name must be “your email id”.cpp

Grading

- We will test your code using the same grading too but for 5 additional examples right after the deadline.

- Your score will be in proportion to the number of examples your code passes .
  - E.g.. 10 / 10 -> 100, 8 / 10 -> 80.

- Please wait up to 1-2 days before you see your scores updated on catcourses

If your submission is overdue

- You will be given another week after the deadline, but you will lose 10%.
  - E.g.. 10 / 10 -> 90, 8 / 10 -> 72
- Your submission won’t accepted after one week after the deadline.
- So, the deadline specified is a “soft” deadline and 1 week + the deadline is a ”hard” deadline

We will strongly stick with the deadlines

- You will typically have 1-2 weeks before the (soft) deadline. You have ENOUGH time to do the hw ahead of the deadline.
- There will be 12 or 13 labs, and 10 labs are enough for you to earn the full score in labs.
- No redemption for violating the soft deadline
- For highly exceptional cases (e.g. hospitalization, family emergency), the instructor may make exceptions. But the instructor reserves the right to decline your request.

Your final lab score at the end

- The lab accounts for 20% of the grading.
- Each lab (from Lab01) is worth 100 pts. You need 1000 pts to get the max out of labs
- Max (1000, Σ_i Your Lab i score) / 1000 * 20%

What if your code works on your computer but doesn't on ours? (this policy is subject to change)

- **Come to your LAB** to speak with **your LAB SESSION TA** (no exception)
- Download **the file you submitted** from catcourses
- **You** must demonstrate that the downloaded file compiles using the standard c++ 14 compiler and pass the 5 test examples you're given
  - If you used a different compiler, we won't do the next, meaning that you will get at most 50% for the lab.
  - Else, **you** must show what changes must be made to make your code run on a lab machine, and send it to the TA; the TA will run your file for all 10 examples
    - If no changes are needed, you will get the full score (possibly minus the overdue penalty)
    - Depending on the amount of changes needed, the TA reserves the right to take off 10-20% of the lab full score.
    - **We expect that you test your code on a Lab machine before submission.**

Academic Integrity

- Read the Academic Integrity Policy carefully

FAQs

Cheating vs. Collaboration.: Collaboration is a very good thing. On the other hand, cheating is considered a very serious offense. Please don't do it! Concern about cheating creates an unpleasant environment for everyone. If you cheat, you risk losing your position as a student in the college. The school's policy on cheating is to report any cases to the university judicial office. What follows afterward is not fun. So how do you draw the line between collaboration and cheating? Here's a reasonable set of ground rules. Failure to understand and follow these rules will constitute cheating and will be dealt with as per university guidelines. The Simpson?s Rule: This rule says that you are free to meet with a fellow student(s) and discuss assignments with them. Writing on a board or shared piece of paper is acceptable during the meeting; however, you should not take any written (electronic or otherwise) record away from the meeting. This applies when the assignment is supposed to be an individual effort or whenever two teams discuss common problems they are each encountering (inter-group collaboration). After the meeting, engage in a half-hour of mind-numbing activity (like watching an episode of the Simpsons), before starting to work on the assignment. This will assure that you can reconstruct what you learned from the meeting, by yourself, using your brain. The Freedom of Information Rule: To assure that all collaboration is on the level, you must always write the name(s) of your collaborators on your assignment in the beginning of your submission file as a comment.

1. **Searching an implementation of the algorithm you're asked to implement.** This is a violation. Of course, there's no way that we can know for certain if you quickly eyeballed other implementations. If you just did, your code will substantially different from what you saw. But the longer you stare at other codes, the more similar your code will be to those. Your risk will get elevated.

2. **Sharing your code with your classmates**. This is a violation. Even if you receive the code only for reference, it will considered as a violation.

3. **Do not post your implementations in public sites (e.g. github) and make them accessible to others.** Trust me. Implementing easy common algorithms won't impress anyone. And your classmates could be tempted to copy them. In particular, if you post your codes before the hard deadline, it will be a violation.

4. **Searching examples of common data structures (different from what you're asked to implement).** This is not a violation. For example, you might have forgot how to use linked lists. In this course, you may need them to implement some graph algorithms.

What Resources Can I Use?

- The textbook (CLRS) includes the pseudocode of most algorithms you are asked to implement. So, you can definitely use them.

- Searching examples of common data structures (different from what you’re asked to implement).

What if you violate the academic integrity?

- After the hard deadline, we identify some potential violations
- The instructor will contact the students suspected of cheating.
- The instructor and the TAs are all certain, the instructor will report your case to the Office of the Student Conduct, and also the School of Engineering, and you will get 0 for the assignment/exam.
- If this is not the first violation for you in CSE courses, the instructor will be notified and your will get F for the course. If you get F for this reason, you are not allowed to drop out of the course
- You can appeal. Then your case will be reviewed by a committee consisting of some CSE faulty.