Assignment Guidelines
CS240E, Winter 2023

1 General requirements

Be sure to check the website at https://www.student.cs.uwaterloo.ca/~cs240e/w23/info.phtml for details on the return policy, remark requests, and academic dishonesty (cheating).

No late assignments will be accepted. If you have the assignment done (or partially done) before the deadline, submit it – you can still change it before the deadline. (Arguments such as "I thought the deadline was today, not 1+ days ago. I had it done on time." when there is nothing in MarkUs to prove that, will not be accepted.) Assignments or assignment components can be excused with appropriate verification of illness and weight distributed over like components.

Files submitted after the deadline can be marked for feedback if the ISAs are notified to look for them within 48 hours of the deadline. (This is highly recommended to test your understanding of material if you miss a deadline for whatever reason.)

The solutions you hand in must be your own work. In particular, you are not allowed to look up the solutions in the literature or on the Internet. An **academic integrity declaration** will be required with each assignment; failure to submit this will result in a grade of 0 for the assignment.

2 Written assignments

What and how to submit: Written work is to be submitted through **MarkUs** as a pdf file. You should always check whether all files you wanted to submit were accepted on MarkUs. Note that you can always re-submit new versions of your files, delete previously submitted files, or submit different parts of the assignment at different times.

MarkUs does *not* permit files to exceet 5M in size (in particular photos of scans are often too big). LaTeX is strongly recommended but not required. The LaTeX-source code of assignments is available from the web page and should get you started. LaTeX-sources for some pictures from the modules are also available on the web page.

Tips for writing your solution: (The following are general guidelines; if the assignment specifies otherwise then follow what the assignment says.)

For questions that ask you to **design/describe/give an algorithm or a data structure**, you should design the best algorithms you can come up with. The first criterion for marking is the correctness, the second criterion is efficiency. Thus, an algorithm that is too slow but correct has a much better chance for partial marks than an algorithm that is fast but incorrect. In your solution, enclose the following:

1. describe the main idea first in words,

2. present all the details (at a level of detail similar to the lecture notes), typically as clearly written pseudocode or as a well written English description,

3. justify correctness (we do not require a full CS245-style proof, but explain briefly why the algorithm returns with the correct answer), and

4. analyze the algorithm (typically by giving a tight bound on the worst-case run-time, but you may also be asked to analyze auxiliary space or other types of run-time).

For assignments typically all four steps must be done, while for exams often only step (2) is required (exam-questions should clarify this, but ask if in doubt).

For questions that ask you to **argue/prove a statement**, you are expected to give a formal proof, demonstrating the series of steps which lead you to the answer. You must justify any "non-obvious facts" that you use in your proofs. *A "non-obvious fact" is any fact which has not been stated/proved in class. In the case that you are using facts stated in lectures/modules, be sure to cite where you got them from.* Be very hesitant to use phrases such as "Obviously...", "It can be easily shown that....", "It is clear that..." because all too often it is not truly obvious (and sometimes it is simply not true).

Typically you should not need mathematical results other than the ones used in the lecture notes or covered in first-year math and CS courses. If you feel that some results from more advanced courses would really simplify the solution, contact the instructor to see whether you are permitted to use it without proof.

Ensure that your solutions are complete and mathematically precise, and at the same time, easy to understand and to the point. Be sure your arguments will convince a skeptical (but intelligent) TA. Your solutions will be judged not only for correctness, but also for quality of your presentation and explanations. If you have are aware that some part of the problem is left unsolved, say so. If in doubt, write down more details (while conciseness is appreciated by the TAs, overlength will rarely be punished).

The course notes have numerous examples of how to describe an algorithm or do a proof; mimic their style.

3 Programming questions

What and how to submit: Programming work is to be submitted through **Marmoset**, and you must submit the program file with the file name specified in the assignment.

Your program should be implemented in C++ on the undergrad Linux environment (`linux.student.cs.uwaterloo.ca`). If you are working on a different platform, it is your responsibility to ensure that your program runs properly on the undergrad environment. Files should be named `<filename>.cpp` , where the specific file name will be specified in the assignment. Skeleton code will sometimes be distributed.

You are not allowed to use code that you found on the web or in existing libraries. As for the standard library (STL), you are allowed to use functions that are simple 1-liners to code

(such as `swap`), and functions/classes that have nothing to do with cs240 material (such as `iostream`), but you are *not* allowed to use functions/classes that implement cs240 material. In particular, do not use algorithms for sorting or searching, or containers such as `vector, priority_queue (unordered) (multi)set, (multi)map` unless specifically permitted on the assignment. You are *allowed* to use `array, (forward) list, deque, queue, stack, iterator` and `pair`.

Tips for coding your solution: Programming questions will usually involve implementing an algorithm or data structure that you have seen in class (or a variation of it). Sometimes details of how to realize parts of the algorithm/data structure are left out; it is then part of your assignment to figure out those details.

You should include sufficiently many comments in your code that the main ideas of design and correctness are clear. Marking will be primarily based on correctness (as determined by our test runs), but marks may also be assigned to the coding style (documentation, design, clarity, efficiency, etc.).

Each assignment will specify the signatures of the methods that you must implement, and how they will be tested. Since we will compile/run/test your program using an automatic script, it is vital that you follow our instructions to the letter. It is a good idea to test your program thoroughly yourself (on your own input) to ensure its correctness. Some pre-testing might be available—this will be announced in the course newsgroup (Piazza).

Good programming practice generally dictates that each class definition should appear in its own file, typically with a header file. For cs240, we will *not* ask you to do this, and you are specifically *allowed* to put multiple class-definitions into one file.

4 Further information

The course web page at https://www.student.cs.uwaterloo.ca/~cs240e/w23/assignments.phtml has much more detailed instructions on how to create PDFs, and how to submit in MarkUs. You should be familiar with Marmoset from previous CS courses; see for example https://student.cs.uwaterloo.ca/~cs146/marmoset.shtml.

If you have further questions about what and how to submit, or about how much information to include, ask the ISA or make a discussion forum (Piazza) post.