CSC263 – Problem Set 2

Remember to write your **full name** and **student number** prominently on your submission. To avoid suspicions of plagiarism: at the beginning of your submission, **clearly state any resources (people, print, electronic) outside of your group, the course notes, and the course staff, that you consulted**.

Remember that you are required to submit your problem sets as both LaTeX `.tex` source files and `.pdf` files. There is a 10% penalty on the assignment for failing to submit both the `.tex` and `.pdf` files, or submitting inconsistent versions of these files.

A Take-Home Interview Exercise

At the beginning of the course, we mentioned that many employers ask data structure and algorithm questions in job interviews. This assignment replicates a method that employers sometimes use to assess candidates, namely the take-home interview assignment. Note that these types of assessments are not without controversy, as they can be time-consuming for the candidate. Our goal is to use this situation as an example of a realistic sitaution so that you can practise explaining your data structure ideas to a professional audience.

To that end, suppose that you are applying to be a Software Developer or Research Assistant at Super-Awesome-Org. The technical team lead at the organization sends you the following email:

> Dear (Your name),
>
> Thank you for your interest in Super-Awesome-Org!
>
> Your resume looks impressive, and we would love to bring you in for an interview. But before we schedule an on-site, we want you to attempt the attached task so that we can discuss your solutions during the interview.
>
> The task is to implement a dynamic hash table that uses linear probing. In other words, we would like you to implement a hash table that automatically doubles in size when it reaches 75%, and reduces in size once it drops below 25% capacity.
>
> There is some Python 3 starter code attached to help you get started. The code contains an incompelete definition of a `HashTable` class, In particular, this class has a `self.array` attribute, which is the address table. There are some additional specifications in this file as well (e.g. that we never go below the initial capacity), and some example calls at the bottom of the file. For this task, please do not use Python's built in dictionary.
>
> In addition, could you also include an analysis of the amortized runtime if we were to start with a hash table with initial capacity of 10, insert N elements, and then delete all N elements? We want to get a sense of how you think about this type of analysis problem, and how you communicate.
>
> Let me know if you have any questions. Otherwise, please send me your completed code, and a written explanation that describes how your code works by March 9th. Thank you!
>
> Riley F.
> Technical Team Lead
> Super-Awesome-Org

What to submit: Submit a PDF file of your email response to this request in the file `ps2soln.pdf` (and `ps2soln.tex`). The email should contain a description of your solutions and a justification of your approach, the amortized analysis that was requested, and a note that the file `csc263_ps2.py` is included as an attachment.

Your email tone should be friendly and professional. It does not need to be overly formal (for example, personal pronouns are appropriate), but should follow the usual format of a professional email:

- Address your reader at the top of the email (e.g. "Hello Riley" or "Dear Riley"), and sign your name at the bottom of the email.
- Make it clear to your reader in the first 1-2 sentences why you are sending this email.
- Use paragraphs to logically separate your thoughts. Each paragraph should have a topic sentence that gives the reader an idea of what the paragraph will be about. Topic sentences are important so that readers can find important information quickly, and to be able to skim through your email.

- We recommend that you end with a call to action, and describe what you would like the reader to do after reading the email. In this case, an appropriate call to action could be to let you know what the next steps are if the organization is interested in moving forward.
- Finally, sign off on the email by including a signature block.

You will be evaluated on not just the correctness and the explanation of your algorithm, but also the quality of your writing. The point breakdown will be as follows:

- (6 points) **Unit tests.** The `HashTable` methods `insert`, `delete`, and `search` will be tested. We will be looking for the correct values of attributes `array`, `size`, and `capacity`. **You must submit and earn points on the written part of the assignment in order to earn unit test points.**
- (6 points) **Explanation and Justification of Correctness.** Clearly explain the approach you used in your implementation. In addition to providing an complete explanations as to how these operations work, please also justify your decisions and show that your approach is correct. We are looking for whether you thought of possible corner cases and handled them appropriately, and whether your explanation aligns with your code. Note that this explanation should not simply repeat your code. Instead, summarize your approach and why it works.
  - Example of a bad explanation: "The code loop through each element of a list, and if there is a 42, and add one to the counter."
  - Example of a good explanation: "The code counts the number of times 42 appears in the list"
- (6 points) **Amortized runtime analysis** You may use a combination of logical and mathematical reasoning in this analysis: i.e. you can use math or words to describe your runtime analysis, as long as your reasoning is clear to the reader.
- (4 points) **Writing: Structure** The email in your pdf file should follow the structure we outlined for a professional email. Ideas should be introduced in a logical order, and the structure should be made clear to the reader. Use paragraphs and sentences, with good transition words between them to guide the reader and let them know how your sentences fit together. Use summaries and topic sentences help readers anticipate what each paragraph will be about. If a reader is looking for something specific (e.g. how you handle one specific case), they should be able to skim your document and find exactly where you discuss that topic, without reading the entire document. You should mostly use complete sentences, though you could also use bullets, tables and figures to highlight important information. An observation should be presented before the analysis that relies on that observation is presented.
- (4 points) **Writing: Clarity/Writing Mechanics** Any technical terms you introduce should be defined. Any pronouns that you use ("this", "it", "they") should be unambiguous—the reader should have no difficulty understanding what pronouns refer to. Write concisely where possible, but not to the point of seeming robotic. There should be few or no grammatical issues distracting to the readers. Opt for simple sentences that introduce one idea, rather than complex sentences that link several ideas together. Keywords (e.g. variable names) should be annotated with their "type". Sentences should be structure to read well "left-to-right", with no new information that changes the meaning of the sentence introduced at the very end.
  - Example of a sentence that doesn't annotate keywords: "We have that `p` is not in `q` unless `i` is a prime."
  - Example of a sentence that does annotate keywords: "We have that the number `p` is not in the list `q` unless its index `i` is a prime."
  - Example of a sentence that don't read well "left-to-right": "The code assumes that (`42 in list`) is not true." (The reader needs to rebuild their mental model of what you are saying when they read the last word in the sentence.)
  - Example of a sentence that does read well "left-to-right": "The code assumes that 42 is not in the variable `list`"
- (4 points) Writing: Audience Expectations Your writing should be appropriately professional, and targeted towards a potential employer, mentor, or collaborator. Avoid slang (e.g. "This code is lit"). Use gender-neutral and inclusive language. Assume that the reader is fairly technical (e.g. has a CS degree), but may not know the exact convention that you are using. Avoid generic verbs like "do" or "make"
  - Example of poor assumptions: "My analysis will assume all properties in slide 35 of lecture 14."

  - Example of good assumptions: "My analysis will assume the universal hashing property."
  - Example of generic term: "The code goes through the list and make each item into a node."
  - Example of technical verbs: "The code iterates through the list items, and construct a node with each item."

Additional Writing Advice:

- As always, start early!
- Your first draft will suck, and that's okay. Write the first draft to clarify your ideas and to get a deeper understanding of the problem. Then, revise your writing so that it becomes more organized and more clear. Good writing always takes several drafts; the first time through is often when you are just figuring out what exactly you want to say.
- When you begin revising, keep the reader in mind. The secret to good writing is the ability to put yourself in the reader's shoes. What does the reader already know? What don't they already know? What might they be expecting to see next? Good writing shows that you care about the reader's time!
- When you write the email, you might be tempted to describe your thought process as you came up with the solution. However, the reader only cares about your final result.
- Use the above guidelines and the rubric as a checklist. Can you replace ambiguous pronouns with the corresponding proper nouns? Do your sentences read well left-to-right? Can you replace generic verbs with technical verbs? (and so on)