Using Chat-GPT to Create Multiple Choice CS Exams

Karol Lejmbach
Computer Science
Marquette University
Milwaukee, USA
karol.lejmbach@marquette.edu

Sean Mackay
Department of Engineering Education
University at Buffalo
Buffalo, USA
0000-0002-0358-7594

***Abstract*—This work in progress Innovative Practice paper presents our work evaluating how Large Language Models (LLMs) can be utilized to aid in the development of scalable and authentic programming exam questions for students in upper-division courses. Traditionally, students' understanding of computing concepts are often verified using written exams. This is especially the case in larger universities due to their overall enrollment sizes, making more authentic, involved assignments particularly hard to facilitate. Since the rise of Chat-GPT and other LLMs, the use of such exams has seen a resurgence in popularity in classrooms globally. Fears have recently grown that programming assignments, particularly those that are take home assignments, are not adequate assessments of students' understanding given the ease at which students can use LLMs such as Chat-GPT to obtain answers.**

multiple-choice exams have been a traditionally popular programming exam format, where-in students are required to choose the correct answer to a prompt or the segment of code that best fits within a larger block of code. Another form of exam question that has been traditionally popular is providing students with blocks of code and ask students to either interpret the code's meaning or find the errors in the code. Both of these styles of assignments provide opportunities for students to demonstrate a deep understanding of code syntax and structure, while also tasking them to demonstrate their understanding of what the intended purpose of the code is. However, the development of these exam problems has traditionally required a substantial amount of work for instructors and teaching assistants to develop. Our goal with this work was to determine if LLMs represent effective tools for developing exam questions. Additionally, we wanted to see if LLMs could effectively design problems based on learning outcomes, allowing the instructor to become the evaluator of the exam questions rather than the original author. Our motivation for this was to develop a reproducible and easier to implement methodology for developing exams at scale for instructors while alleviating the workload imposed on instructors during the development of exams.

Our initial research has indicated the use of these LLMs for exam problem generation greatly reduces the workload for instructors while allowing for the creation of far richer programming questions for students that require them to apply more of their knowledge to individual problems. We have had a good amount of success in developing these problems for upper-division courses, as well as introductory-level courses. This work represents initial steps towards the use of these LLMs for generating exams and more work is needed to determine the actual efficacy and long-term benefits and reliability of these tools. Regardless, we are confident that LLMs in their current form represent incredibly powerful tools for instructors to utilize in the development of their course exams.

***Index Terms*—large language models, exam development, computing education, Chat-GPT**

I. Introduction

In the past two years, there has been a surge in interest around the development of Large Language Model's (LLMs). This surge peaked with the release of open-AI's Chat-GPT. Since its release, there has been growing concern among educators about its use in classrooms. While some have argued for allowing the use of Chat-GPT in course environments, others have argued that the use of such tools will lead to students using such tools disingenuously [7]. Some people within computer science in particular have argued that with the advent of these LLMs, being able to effectively evaluate student understanding has become substantially harder. Our position as the authors of this research has been mixed on this issue, with disagreements over the use of Chat-GPT in classroom environments being a recurring topic. However, we have come to the agreement that given a course environment in which students have access to Chat-GPT for use with take-home assignments or projects there is no clear and accurate method of evaluating whether work has been generated by the student or by Chat-GPT.

Bloom's Taxonomy defines a variety of levels of student understanding of a concept [2]. In traditional education contexts, there has been a focus on evaluating students on lower-order levels of their understanding such as their ability to recall facts or explain concepts. Computing Education Research (CER) has traditionally focused on developing course assignments that instead test students on their ability to create or produce new or original work. These sort of programming assignments are excellent at evaluating students higher order levels of understanding of programming concepts. However, when considering the prevalence of Chat-GPT and the lack of any method of determining what code has been generated using Chat-GPT, there has been an increased interest among educators for the use of exams as a primary means of assessing student understanding [7]. While exams do not assess students' ability to create new material, a well designed exam is capable of satisfyingly analyze and evaluate, the two levels below create on Bloom's Taxonomy [2]. While these exams do provide instructors more assurances about whether students

are completing their work honestly, the development of these exams in a scalable and easily gradable manner has routinely proven to be a daunting task for instructors.

The work presented in this work in progress has been conducted in an effort to determine the viability of Chat-GPT and other LLMs for the development of exam questions for a range of courses and environments [4]. Our goal with this work was to answer the following research question:

1) Can LLMs generate meaningful multiple-choice questions for Computer Science exams?
2) Do these questions provide satisfactory means of evaluating student understanding of the course topics?

Our work will present the findings from these initial attempts at generating these multiple-choice fill in the blank programming problems, as well as results from our initial attempts at evaluating these problems. Our initial findings have been positive, and we will discuss these findings in section four below. While we feel as though more work is needed, we are confident that Chat-GPT and potentially other LLMs serve as valuable resources for instructors as they go through the process of developing exams for larger courses, particularly because of its ability to generate variations on a problem of comparable structure and difficulty. We also argue that the use of Chat-GPT for producing these exam questions also allows for the generation of unique exams for each student, allowing for opportunities for more equitable exam designs such as allowing students more flexibility for makeup exams. This allows for the opportunity to support students in a number of ways that will better help them succeed when completing these exams. It also allows for instructors to have fewer concerns about students' honesty when proctoring exams. Additionally, these style of exam questions, we argue, are more representative of how students will be expected to utilise their knowledge in a professional environment, as writing code from scratch is not normally a task that is assigned to new hires. Rather, newer software engineers, as well as anyone working with code, is often tasked with working with, editing, debugging, and modifying code developed by others.

II. Background

Exams and programming projects or assignments are traditionally popular methods for evaluating student knowledge. There has been a wealth of research into the development of both of these assignment types for a number of years. In terms of programming assignments, there has traditionally been a substantial focus within the CER community into developing methods for automated assessment for programming solutions [6]. A common issue that has been discussed with automated assessment for grading programming assignments is the lack of coherent feedback for the learner. Recent research has tried to resolve some of these issues by investigating the usefulness of LLMs as a method of providing better feedback to students about their submissions without manually providing this feedback [1]. This research shows that there exist instructional benefits for the use of LLMs and has worked to improve the level of feedback provided to students by these automated systems [6].

While this work does provide solutions to better improve the experiences of students as they approach assignments within a course, it does little if anything to improve the methodology used for the development of exams. Since the development of LLMs, there has been a growing concern regarding the efficacy of take home assignments [7]. We do not aim to take up a stance with regards to how the use of LLMs should be handled within courses; rather, the goal of this work is to show its benefit as a means of supporting instructors in the development of exams.

Regardless, the concerns felt by many across computing education space underscores a continued demand for exams, and therefore a need for methods of supporting instructors in the development of these exams, as well as means for ensuring exams are fair and meaningful assessments for student knowledge. Research focusing on exam development has traditionally focused on understanding what impacts students' performance on exams, as well as innovations in exam formats such as computer-based or online exams [3]. One abstract for a paper coming later in 2024 that has already been published online has already begun the process of considering the use of LLMs for the development of exams, though their work according to the abstract has focused specifically on exams for an introductory python programming course [5]. While this abstract shows no findings, it underscores a growing interest in the use of LLMs to aid instructors, and shows a potential positive perspective on LLMs in higher education.

Besides this, there has been little research into the development of tools to aid instructors in the development of scalable exams, particularly those using LLMs. This is due in part to the relatively new nature of LLMs as a force within education and the world as a whole. Our research aims to work towards filling this gap, while exploring a different group of courses other than first year introductory courses: upper-division computer science courses.

III. Methods

A. Exam Structure

The research in this work in progress consists of initial investigations into the development of problems for two courses that typically represent core topics within the curriculum: Data Structures and Programming Languages. In order to develop these initial questions, we approached this work from the core learning objectives from each course. After isolating the learning objectives from each course, we designed and tested prompts for Chat-GPT. Responses to prompts were then evaluated by the other author to determine how closely they aligned with evaluating the learning outcomes that had lead to the prompts. Our goal was to focus on the development of multiple-choice exam questions for both courses, as both courses at one of our respective universities utilizes multiple-choice questions for these courses' exams.

B. Large Language Model Prompts

Working with Chat-GPT posed a few interesting challenges. Our intention was to let Chat-GPT generate everything from a single prompt; however, the responses we received required additional clarifications and requirements to refine the questions. The first challenge was figuring out if the prompts we were providing gave sufficient information to generate useful exam questions. Since Chat-GPT was generating both the question and the multiple-choice answers, we had very little control over how it would phrase the question as well as what choices it would provide. When prompted to create Data Structures related questions, regardless of data structure, it gravitated towards Big-O analysis-related questions and we had to explicitly ask it to generate questions not related to Big-O.

The second challenge was ensuring that the multiple-choice options had only one true answer. Chat-GPT-generated choices had overlap, making the correct choice harder to isolate given the question. For example, Chat-GPT would often provide options wherein one was fully correct, but another option was partially correct. As part of our testing we would ask Chat-GPT what it thought the correct answer was. In the scenarios with more than one correct answer, we asked about the second choice. In those situations Chat-GPT would acknowledge the nuanced differences, but would still point to its first choice as the correct answer.

IV. Initial Findings

In this section, we highlight some of the more interesting initial findings from our research into the use of Chat-GPT, as well as other LLMs in the development of multiple-choice exam questions for Data Structures and Programming Languages. We begin discussing examples regarding data structures learning objectives and concepts, then discuss two questions from programming languages that we developed utilizing Chat-GPT.

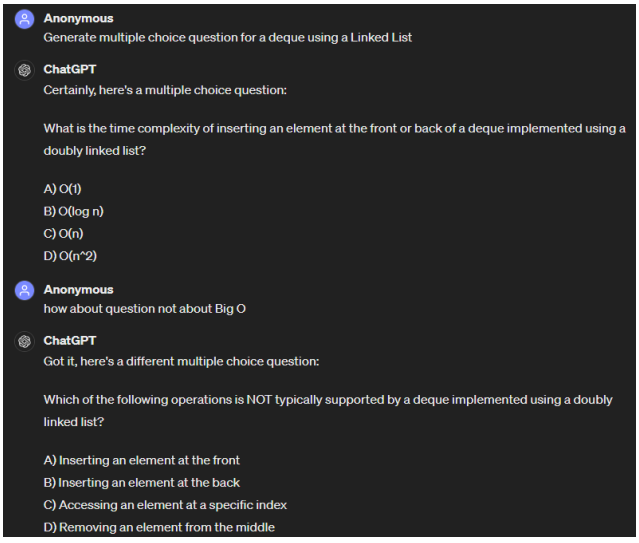

Fig. 1. multiple-choice questions about Deques using Linked Lists

Figure 1 is a screenshot taken from one of the sessions where potential multiple-choice questions were generated. It is worth noting the second question generated. Both choices C and D satisfy the prompt and when asked Chat-GPT identified C as the correct answer. When asked about D, it did explain that it too could work as a potential answer to the generated question.

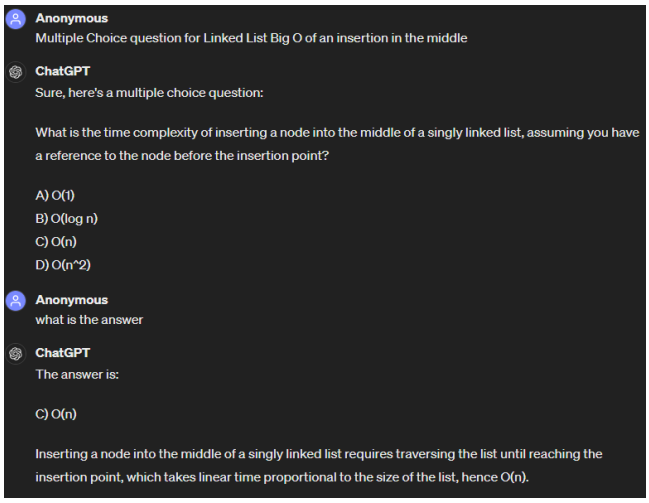

Fig. 2. Chat-GPT generated question with wrong answer marked

Figure 2 demonstrates that Chat-GPT does not always identify the correct answer. Based on the scenario described where the location of the insertion is all ready known it should take O(1) time.

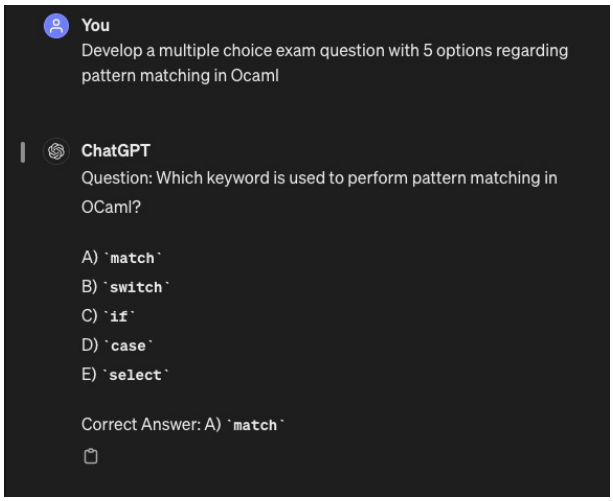

Fig. 3. Correct Ocaml question regarding types

Figure 3 is an example of a question that we generated for a language-specific question for a programming languages course. Programming Languages at one of our universities has conventually covered OCaml as being central to the course structure. Typically, these exams have consisted of multiple-choice questions and have posed a substantial lift for instructors to accomplish. Chat-GPT was able to generate this,

as well as several other questions regarding specific concepts and learning objectives and managed to accurately produce multiple-choice questions for them.

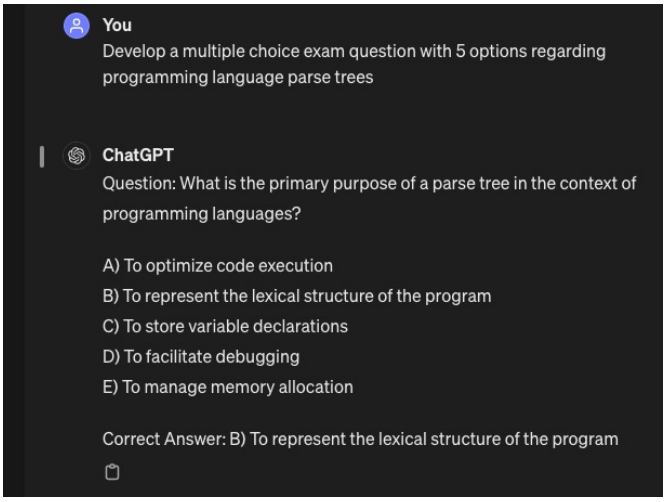

Fig. 4. Parse tree question illustrating issue of ambiguity

Figure 4 is an example of a question generated asking about parse trees in programming languages. The result of this prompt highlights one of the potential issues presented by LLMs for the development of exam questions. The multiple-choice options given have a level of uncertainty to them that is perhaps unique to LLMs. For example, while option B is correct in that the primary purpose of parse trees is generally to represent the structure of expressions within code, parse trees can also been seen as potentially valuable tools for code optimization or debugging depending on the context in which they are presented. This highlights one of the potential limitations of Chat-GPT and other LLMs within the development of exams.

V. Discussion

This initial work that has been conducted has indicated that there are potential benefits to using LLMs to generate exam questions. The first major benefit is the role change of the instructor from being an author of the exam questions to an editor. This change in roles makes proof-reading an exam much more effective. Additionally, the use of LLMs allows instructors to pivot back to developing exams from a learning outcomes-focused lens, as prompts can be generated directly from learning outcomes that need to be assessed. While this is typically considered a beneficial methodology for question development, it is typically not the norm for exam development.

Another major benefit we see is the easing of the logistics when it comes to courses with large enrollments. It is common practice to provide multiple versions of an exam to mitigate student copying. By generating exam questions using LLMs more versions of the exam can be created with less effort. This means that instructors could develop 4 or 5 versions or more of an exam in much the same time as it would typically take them to develop 1 or 2 versions.

Through our initial work, we have identified some potential drawbacks to LLMs for exam development. The biggest challenge is ensuring that the LLM has an accurate understanding of the material. For well understood topics this is less of a problem; however, with more obscure concepts found in more specialized classes the accuracy of the questions may be questionable, and will need further evaluation by instructors to ensure the accuracy and fairness of the questions. This is why the instructor still needs to evaluate the questions and make edits to ensure the questions are worded carefully and without too much room for confusion by the student.

Another challenge that can come into play is question repetition. For sufficiently large exams there exists a possibility depending on the concepts being evaluated that Chat-GPT or other LLMs start producing repetitive responses to prompts. In this case a potential solution would be refining the prompts for the LLM to improve the spread of topics, and is something that is worth considering in future research in this area.

From this preliminary work, we see that there is strong potential for LLMs to be useful for instructors within upper-division computer science courses. Implementing LLMs for the production of exams for these courses will require more refinement and thorough analysis to insure the accuracy, fairness, and validity of the questions that different prompts produce. At the same time with additional improvements to LLMs, the process of generating these forms of exams could yield individualised exams for students, while potentially reducing the overall workload for instructors.

VI. Future Work

We have seen that LLMs can create non-trivial multiple-choice questions. The next step in our research will be going through and identifying which courses these types of exams are appropriate. This will involve piloting the use of these LLMs in the development and dissemination of exams within different course environments. This work will allow for the discovery of whether or not LLMs are beneficial during the development of actual course exams, which courses benefit the best from the use of this technology, and to what level can these tools be scaled.

Another feature worth exploring in these exams is LLM-generated code. Students will often be expected to work with and debug existing code both in their academic and professional lives. The skill-sets required for debugging existing code are far different than debugging their own code. These types of exam questions can help evaluate students' comfort levels and abilities with this skill. This serves as another potential avenue where LLMs could aid in the development of exam questions that have more authentic random code for students to evaluate and debug.

References

[1] Rishabh Balse et al. “Investigating the Potential of GPT-3 in Providing Feedback for Programming Assessments”. In: *Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 1*. ITiCSE 2023. Turku, Finland: Association for Computing Machinery, 2023, pp. 292–298. ISBN: 9798400701382. DOI: 10.1145/3587102.3588852. URL: https://doi.org/10.1145/3587102.3588852.

[2] Benjamin Samuel Bloom. *Taxonomy of educational objectives: The classification of educational goals: Handbook I, cognitive domain*. McKay, 1969.

[3] Anthony Estey and Yvonne Coady. “Study Habits, Exam Performance, and Confidence: How Do Workflow Practices and Self-Efficacy Ratings Align?” In: *Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education*. ITiCSE ’17. Bologna, Italy: Association for Computing Machinery, 2017, pp. 158–163. ISBN: 9781450347044. DOI: 10.1145/3059009.3059056. URL: https://doi.org/10.1145/3059009.3059056.

[4] *Introducing ChatGPT*. URL: https://openai.com/blog/chatgpt.

[5] V. Juričić and M. Obrvan. “STUDENTS’ PERCEPTION OF AI GENERATED EXAM QUESTIONS”. In: *INTED2024 Proceedings*. 18th International Technology, Education and Development Conference. Valencia, Spain: IATED, Apr. 2024, pp. 1480–1487. ISBN: 978-84-09-59215-9. DOI: 10.21125/inted.2024.0435. URL: https://doi.org/10.21125/inted.2024.0435.

[6] Sidhidatri Nayak, Reshu Agarwal, and Sunil Kumar Khatri. “Automated Assessment Tools for grading of programming Assignments: A review”. In: *2022 International Conference on Computer Communication and Informatics (ICCCI)*. 2022, pp. 1–4. DOI: 10.1109/ICCCI54379.2022.9740769.

[7] Parsa Rajabi et al. “Exploring ChatGPT’s Impact on Post-Secondary Education: A Qualitative Study”. In: *Proceedings of the 25th Western Canadian Conference on Computing Education*. WCCCE ’23. Vancouver, BC, Canada: Association for Computing Machinery, 2023. ISBN: 9798400707896. DOI: 10.1145/3593342.3593360. URL: https://doi.org/10.1145/3593342.3593360.