

Faculty of Applied Sciences  
Bachelor of Science in Computing

**COMP490 Final Year Project  
Workbook**Academic Year 2022/23

|  |  |
| --- | --- |
| Parsons problem generator and solver | |
|  |  |
| Project number: | 19 |
| Student ID: | P-19-0834-5 |
| Student Name: | Jane Liu |
|  |  |
| Supervisor: | Philip Lei |
| Assessor: | Charles Lam |

# First semester W1-W14

This section records meaningful discussions with supervisors, consideration and decision processes within project design and implementation, difficulties encountered, and solutions/workaround taken during the first semester. This section aims to provide some ideas for the final report.

## Week 1

Report Date: 26/08/2022

**Meeting Record:**

About the proposal:

Explain the scope and determine what is going to do. The related work can be used for comparison to show the strengths of your projects. The schedule should be reasonable, meaning the work should be done on the weekdays, not weekends. We can convert the weeks to days. For risks, the risk during the developing period and the launching period should be considered.

About the ranking:

Rank A: need to have a new idea, new experience, not just limited to the existing work

Rank B: apply for previous experience/current work, user-friendly platform, explain why

About the project:

The teachers can enter the questions. The generator can cut the question into a block (for example, cut every line into a single block). How to miss the block. The student drag-and-drop the block to the blank positions. And the system can give some feedback(right or wrong). For this project, if you want to get A, you need to do something new not only for the introductory level but also for data structures and algorithms (for example)

## Week 2

Report Date: 31/08/2022

**About the proposal:**

Project Number?

Assessor?

Do we need Gantt Chart in project work plan?

Is the traditional code writing style relevant work?

Which content should cover?

Project description: project problem, why important, objectives, expected results/outcomes, difficulties/challenges

Summary of related work and key references

Project work Plan: tentative schedule (key deliverables)

Risk Assessment: key risks, contingency plan

Ideas from the proposal:

Project Description:

Present situation and the limitation of the existed solution (illustrate project problem and why important) (it can be “It is a trend to learn programming, but the beginners meet a lot of difficulties in traditional code-type writing and they are easy to lose the interest.”)

The aim of the project: summarize what the project will look like in high-level (it can be “design and build a website for providing different levels of Parsons problem in Python for students majored in computing to practice”)

Objectives: it can be the major steps of this project, for example, background study and related work to get ideas about how to improve,

* design and develop a database,
* let the authorized users like teachers to upload the questions and correct code,
* cut the code appropriately,
* mix the correct parts of code with wrong ones (disordered or some common mistakes) (can choose different variants of Parsons problem?),
* design and develop a webpage to let the students to drag or drop the correct answers from the answers pool to the appropriate positions,
* evaluate the problems and give the feedback, which means that the platform can actually run this code and show some error messages with some comments

The strength of this project? Explain why can achieve the requirements? (expected result) Explain difficulty

(new page)

Related Work:

Summarize?

Are there relevant project? Brief description about the functions of these relevant projects? What is the drawbacks of these projects? What will my project achieve? What is the advantages compared with the relevant work?

Project Work Plan:

Summarize?

Excel for the project

(new page)

Risk:

Summarize?

What are the risks?

Computer malfunction -> GitHub

User cannot understand well -> HCI

Different device -> responsive design

Server crash -> distributed architecture

Table of Priority Risk? Explain why.

Probability of the risk (before and after) brief explain

Ideas from the reference paper:

Pre-scaffolded vs Student-scaffold difference?

Need to run the code written by students?

How to cut reasonably? Cut the part including the intentation?

(Thus, it is advantageous for this project to introduce a new method for learning programming, Parsons problem.)

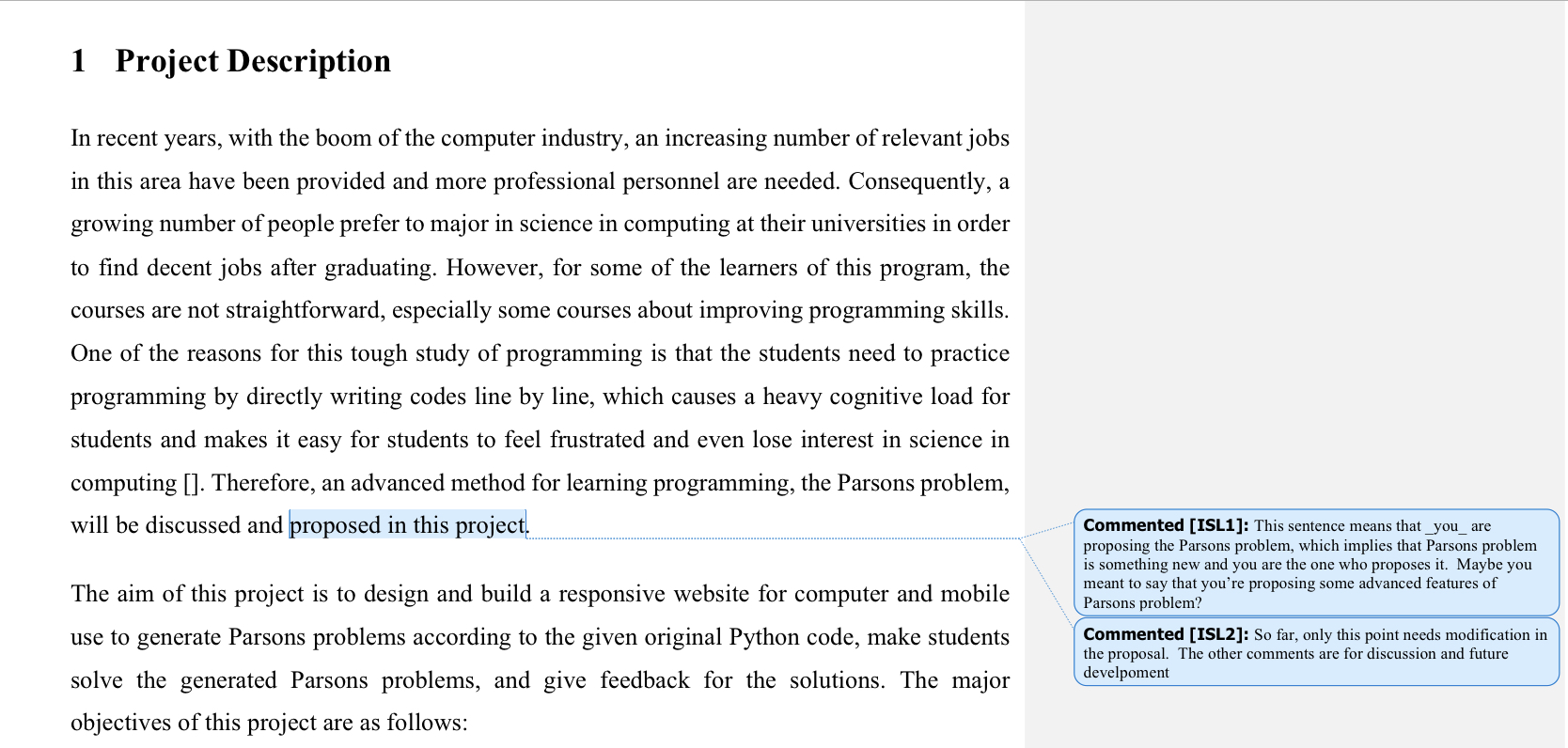
(reordering the mixing up prepared blocks of the solution)

(Security of database)

Report Date: 01/09/2022

**Meeting Record**

About the project proposal:



Text, application

Description automatically generated

Project Number and Assessor will be given next week

Gantt Chart can be written, but not necessary.

The first paragraph of the project description needs to be changed since it may cause some misunderstandings for readers. It should be illustrated clearly what Parsons problem do and what your project do beyond the existing Parsons problem. You can write that Parsons problem solve the previous problems caused by the traditional programming style. But the existing Parsons problem have some limitation (e.g. only in an introductory way, not cover the middle level such as Data Structures and Algorithms) so your project has further development, which shows that the value of your project not the value of Parsons problem.

For the relevant work, you can firstly illustrate different ways in computer education like programming trace. And then focus on the previous Parsons problem to find that what they did and what limitations they have. Finally, illustrate why is needed for your project.

About the project:

Distractors: this project should connect the correlated distractors with the specific blocks of code instead of the whole solution. It means that it can allow the teachers to input or upload the distractors only about some part of the code not all the solution codes.

Run the student code to check? OK, it may not be difficult to run Python code since there are some platforms to achieve the function (Need more information). And it is advantageous since it is more flexible to check the code when it does not matter that some orders of the code are changed (like a = 0; b = 0). There are some ways to evaluate the running result, for example we can compare the actual outputs with the standard outputs, and we also can let the code to print some value for the code without outputs (like the text programming in the interview of TikTok). We can let the teachers to enter the outputs, but the teachers may also have errors. To provide this situation, we can also run the teachers code, and give some alerts to the teachers when the errors happen.

For evaluate the students code and give feedback, we can use the error message reported by the previous running system. Or we can also compare the students code with the answers line by line and mark the correct code and wrong code in different background colour. Or we can use some comparing programs to shows that how many changes are needed and then tell the position of needed changes in the next step. It means that the first tip is to tell students the number of changes like adding, deleting and altering positions, and then let them to think a little bit, if they need further help, the second tip about the exact changes position will be given.

Pre-scaffolded vs Student-scaffold difference? Pre-scaffolded keeps the structure of the program. For example, for the for loop, we do not break the for loop structure (for loop indentation), and we just change the order inside of the for loop and outside separately, which means that we do not ignore the for loop structure and mix the code never mind outside or inside. And another method is to ignore the structure and just mix everything without caring about the levels of code. In that case, student should also set up the indentation.

How to cut reasonably? Cutting it in line is more appropriate since if we consider the whole for for loop, as an example, as a unit, it will be in top-level and not so fit for the middle level students. We can allow the teachers to cut part of the code or the whole code, so that the teachers can focus on specific parts.

Related work about proposal:

[Code tracing: Code tracing is a method in which the programmer uses paper and pencil to hand trace the execution of a program or code segment in order to track the variable values as they change during execution and to determine the output of the code.](https://microcredentials.digitalpromise.org/explore/code-tracing)

Code writing:

## Week 3

Report Date: 05/09/2022

**Related work of project proposal:**

[Coached Program Planning: Coached program planning is a dialogue-based style of tutoring aimed at helping novices during the early stages of program writing. The intent is to help novices understand and solve problems in their own words through the construction of natural-language style pseudocode as the first step in solving a programming problem.](https://dl.acm.org/doi/10.1145/611892.611955)

Another different point, execution feedback?

Need to read the previous writing part to have a better logic with in context

**Project:**

Separate the whole areas into several steps according to logic?

Differences in developing tools in Data Structures and in introductory programming

Look like a game (ppt in Bologna)

Check the objective to add some, like personal page, teachers can check some report got from students

Report Date: 07/09/2022

**Meeting record:**

For the proposal, in the second paragraph of the summary of related work and key references, “drag and drop prepared blocks instead of writing codes” is ok, but it will be better to add some detailed characters of Parsons problem, such as reducing the need of syntax, giving some ideas about some solution structures for some questions, mix the code reading with the code writing, achieving some function from code tracing, give a good example for variable naming and so on.

For the proposal, in the third paragraph of the summary of related work. It just describes the common advantages of using Parsons problem in introductory programming and Data Structure. But it does not explain the specific advantages only for Data Structure. What is the difference of using Parsons problem in introductory programming and Data Structure. For example, it may let teachers use Parsons problem flexibility to test the understanding of some code order. It needs to compare the features between introductory programming and data structure. It needs more information supported by papers.

For the project work plan in the proposal, the detailed chapters can be removed in the plan. Besides, it needs to have a test stage.

For project, it is ok to give subgoals for students (separating the whole solutions into several parts according to logical steps). It is a kind of pre-scaffolded.

Next week's work: read previous work to know the structure of Parsons problem, how to set distractors, and so on. And also find some previous work in using Parsons problem in data structure to know the way they use.

## Week 4

Report Date: 12/09/2022

**Why do we use Parsons problem in Data Structures and Algorithms? (some possible ideas)**

1. To avoid repeating some tedious basic programming syntaxes again and again, for example, for-loop headers. In this way, it can help reduce the possibility of causing some careless mistakes in these basic syntaxes so that students may not need to waste a lot of time figuring out these mistakes and they can more focus on core parts of problems about understanding Data Structures and Algorithms. It looks like do not allow students to use calculators in the math exam in their high school, but allow students to use calculators in university. Parsons problem is the “calculators” in programming.
2. We can use distractors to compare and show different data structures quickly. For example, we can fix other parts and only show the distractors which need to be compared. Teachers can drag and drop these distractors blocks during class quickly to show the differences instead of taking time to type the code line by line. Also, we can set the distractors for different methods of the same problems such as depth first search or breadth first search in tree traversals. In this case, the distractors are not only used for showing some incorrect or improper syntax, but also comparing different correct and proper syntax.
3. Parsons problem may not only be used to rearrange the solution code, but also some structure diagrams, such as the data structures of the result values or the result picked points in Dijkstra’s algorithm. More like code tracing… But structure diagrams of variables instead of the value of variables.

**About Project:**

Game-style interfaces: Gamification ppt from the University of Bologna. Game Design Elements: Points, Levels, Achievements, Badges, Rewards, Leaderboards.

Report Date: 14/09/2022

**For Parson’s Programming Puzzles: A Fun and Effective Learning Tool for First Programming Courses:**

The Parson’s Problems follows the five core principles. To be more specific, the problems should maximize the engagement with puzzle-like game-style questions, constrain the logic, permit common errors by setting distracters, model good code and provide immediate feedback.

Puzzle setting structures: drag-and-drop style, code fragments can include single or multiple lines, and some subsets of these code fragments can constitute the solution. Click the check button to get feedback. Repeat to 100% correct. The problems can be described in text or activity diagrams. And activity diagrams have more advantages than simple text since students need to read and understand the diagram, in other words, they need to understand the solution logic of problems. To some extent, it seems like to add the method Coached Program Planning in this Parsons Puzzles. For distractors, these components can not only be used to point out some common mistakes but also train some good habits, like using the declared constants, identifying the correct keyword, formatting rules, variable names, and even proper indentation.

Implementation: [Hot Potatoes](https://web.uvic.ca/hrd/halfbaked/)

Future work: different problem types: multiple choice, fill-the-gap, and crossword. Increasing the problem pool -> set up questions for each other. It may not be a good idea, since the teachers are more professional and they can set more significant distractors. Besides, the teachers can check frequently mistakes and explain them in the classroom.

Enhancing user feedback -> impossible to predict every potential error -> give an explanation only for the correct version.

Track their long-term progress, showing the number of errors they make for a given topic over time.

Enhanced User Interface: lack of sound, animation, and color. More stylish interface, incorporating attractive graphics and easy navigation between topics. animation rewards for completing puzzles. Like the small rocket in PythonAnywhere.

Enhanced Data Collection: keep detailed records of which items are chosen during each iteration of the puzzles.

**For Two-Dimensional Parson’s Puzzles: The concept, Tools and First Observations.**

2D parson’s puzzles: The vertical dimension is used to order the lines, whereas the horizontal dimension is used to change control flow and code blocks based on indentation as in Python.

Insert a line between two adjacent lines of existing code. (not supported in Hot Potatoes)

Python, a website where teachers can browse existing puzzles, and create collections with several puzzles for their students.

Different types of puzzles: extra lines (group the distractors with the correct alternative and always keep them next to each other in the initial random order -> reduce the cognitive load), User-created blocks (insert curly braces or indent the code), Context (provides a fixed code around the code to be sorted. It allows larger, and often more concrete, examples to be shown to students)

Tools: ViLLE for Java, CORT

Mark the correct/incorrect position in different colors

How to write background (critical analysis of existing works?) Compare with my project? Explain the components in Parsons problem

design approach(high-level description of the methods): a brief overview of the tools and methods that you will use to solve the problem; Briefly describe the interface or the details design of every function like the distractors can be connected with the correlative blocks instead of being put randomly. Database Design? Activity Diagram?

Report Date: 15/09/2022

**Meeting Record:**

**For the previous ideas about how to use Parsons problem in Data Structures and Algorithms, the supervisor has the following suggestion:**

The first one, for avoiding repeating tedious code structure, is not so specific for Data Structures and Algorithms since it also can be used in the introductory programming course.

The second one, for using distractors in showing different correct answers, is worth further thinking. We can use distractors and the original programming code fragments to mix different solving methods for the same problems. For example, for sorting numbers, we have different sorting algorithms such as bubble sort, selection sort, insertion sort, etc. They are both correct, but they are all similar and confusing for students. When students only learn one of them, they know it well. However, when they learn several similar algorithms in the same categories, they feel confused and use some parts of one algorithm and other parts of another algorithm at the same time because of blur memory. In this case, we can set different levels of distractors. In level 0, we have no distractor and only test whether the students know the specific algorithm. In level 1, we add a set of distractors of another similar algorithm to test whether students can figure them out, strengthen their memory, and make them understand the algorithm better. In level 2, we can add a set of other distractors again, and so on. This idea also can be used in other questions, such as searching, only if the questions have several solutions. In these questions, we have more than one answer. The student should be marked as correct if they provide one of the answers with match code fragments.

The third one, for using Parsons problem in code tracing, is good but is a little far from Parson Problem. It can be implemented in the second stage after finishing the major parts. But the supervisor improves a new idea based on this idea. We can provide an original tree of a variable, for example, with another tree of this variable after some changes in the positions of some nodes. And then let students rearrange the code fragments to get the output tree structure. This kind of question is quite suit for the “order code fragments” property of Parson Problem. And in this question, students can practice some basic skills in learning Data structures and algorithms such as building nodes, connecting nodes, and rearranging them.

**What needs to do in the future?**

Learn about the question types in Data Structures and Algorithms to think about the specific usage of Parsons problem. The purpose of using Parsons problem is in particular data structures and algorithms questions like distinguishing different methods and manipulating the data structures.

The questions should be designed as specific questions so that students cannot solve these problems by searching the general solution directly on the internet.

Design Parsons problem with the purpose to strengthen students' some parts of basic skills in data structures and algorithms by giving some input and output.

**For the final report:**

Chapter 2: In this period, we can just write as reading notes of papers, and add more information in the future. No need to describe every detail of Parsons problem in this chapter. In this chapter, the major thing is to describe which area the existing work is good at (strange word, but guess you know the meaning), and what the existing work is not good at. The existing work certainly cannot work well in every situation. Thus, in this chapter, it is important to point out, what is the scope that the existing work is limited to, what is an empty area for existing work, and what is the gap in the existing work. That is the key point that why we need to implement our new work.

Chapter 3: No need to put every related work in chapter 2. The most important thing is to let readers easily understand. To achieve this purpose, we can also pick up our new ideas by comparing the existing work. This part is about designing. For example, we can put our idea about how to use distractors in data structures and algorithms here. If we use experiments to test the effectiveness of our project in the course of Data Structures and Algorithms, we can focus on illustrating the methods of the experiments instead of the design for project implementation. It all depends on what we think is more important.

Chapter 4: It focuses on implementation. All the programming code should be put in chapter 4.

## Week 5

Reported Date: 19/09/2022

**Different types of questions in Data Structures and Algorithms**

To sum up, we have the following question types:

1. Define a class for a specific data structure => Similar to introductory programming. Worthless to use Parsons problem in this kind of question.

Text

Description automatically generated

1. Manipulate Data Structures: May be more suitable for filling in the blanks, cut code in a line into smaller fragments, but what is the difference between this Parsons problem with code writing?

Text

Description automatically generated with medium confidence

Graphical user interface, text, application

Description automatically generated

We can use a series of graphics to give students hints to reorder the code, but it may be too easy. Only the original graphic and the last one? Splitting a List also is an example.

A picture containing graphical user interface

Description automatically generated

Diagram

Description automatically generated

Diagram

Description automatically generated

Diagram

Description automatically generated

Graphical user interface, application

Description automatically generated

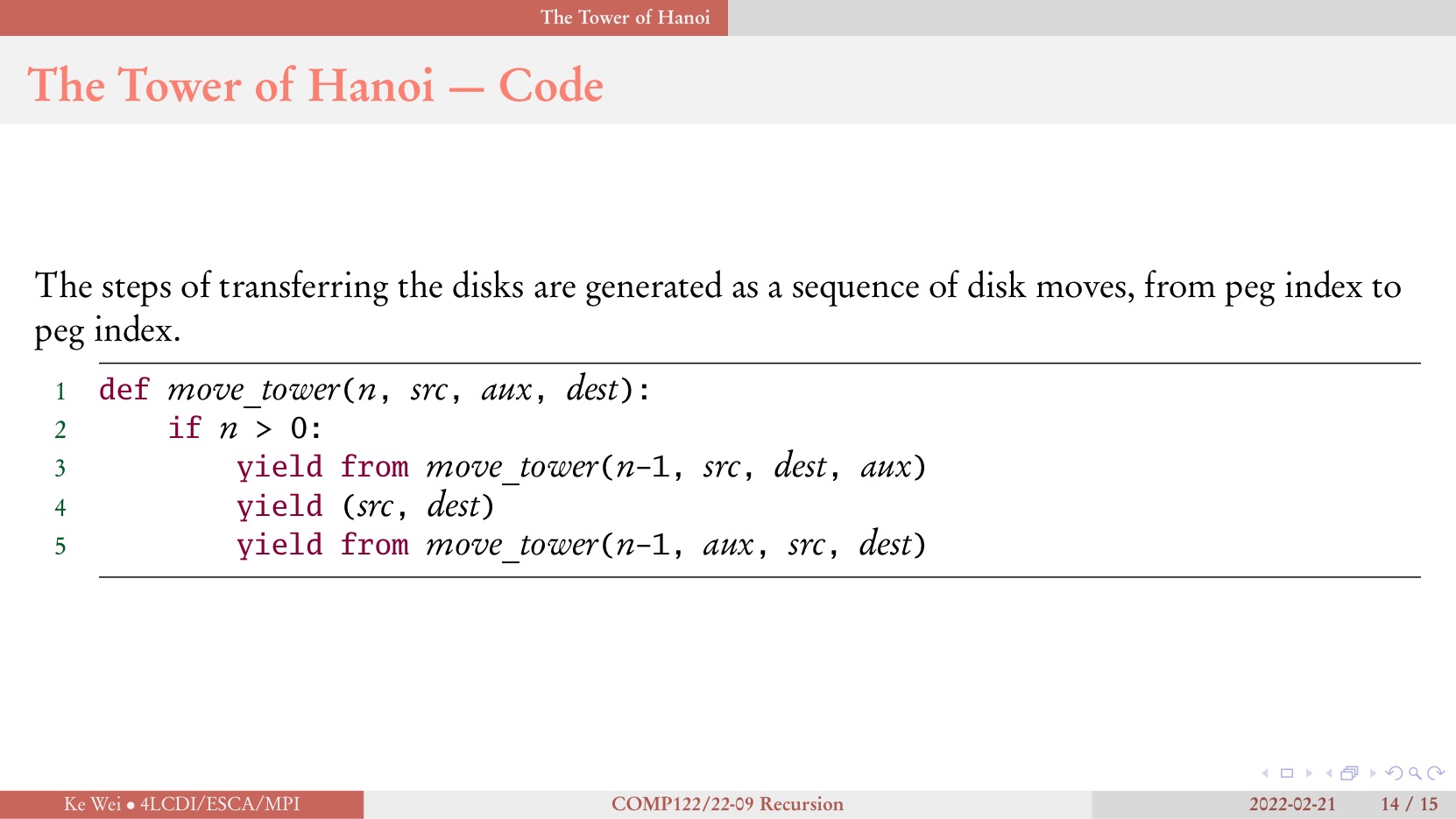
And after learning recursion, although the procedure shown on the diagram is quite complex, the code is quite simple. Thus, the teacher may ask students to draw the diagram to test students’ mastery level instead of coding. If we do not use the recursion, the solution will be too tedious, although it can make good use of the reorder property of Parsons problem.

Graphical user interface, text

Description automatically generated

A picture containing background pattern

Description automatically generated



1. Mathematical Induction & Algorithm Analysis => Mathematical Issues => Ingore

Report Date: 21/09/2022

**Questions in proposals:**

The objective to describe the database needs to be changed later. The including examples are not on the same level, which means that the scope tags of problems should be included in the generated Parsons problem. And the objectives need to flow the SMART requirement. “Generated Parsons problem” is so strange! The problem is not generated but inputted! Delete the redundant “Design and develop a webpage”

**Questions for the meeting:**

Any improvement in the proposal?

Do we need to submit the weekly status report and workbook every week?

Problems in actual questions used in the algorithm course.

Can I record the meeting by digital voice recorder?

Use feedback or not? Can the same feedback be hold by multiple users?

Save solution or not? How to save the solution/record? Save in a file?

**Questions in documents**

ER diagram, Mandatory or Optional? “Appertains” Reverse?

## Week 6

Report Date: 22/09/2022

**Meeting record:**

**For using Parsons problem in actual specific questions:**

For manipulating data structures: The example about duplicating the elements of stack s is a good example. This example is not so general that students can search it online directly, and this example accords with what the supervisor talked about last time well (give input and output, let students learn how to build the basic data structure). For this question, it is not so proper to cut the code in a line into smaller blocks, since in this way, the boundary between Parsons problem and code writing is blurry and it does not look like a Parsons problem (Not provide more information to students, and not reduce the cognitive load). Thus, we still cut the code into lines instead of smaller parts, but we can provide some incorrect distractors to let students choose from them to achieve the results of cutting it in smaller parts (For example, s.push(a.top()), s.push(a.pop())). This question is also meaningful for students to practice some basic skills and knowledge such as FILO in a stack. And for this kind of question, we may use a pre-scaffolded structure to group some codes as a larger block (Taking duplicating the elements of stack s as an example, we can divide the whole code into four parts – every while loop as a part, and the code inside in a part cannot be moved to another part so that the code can just change order or change indentation inside the part) and fix some code fragment and let students change the others (like in the joining two lists, fix the first two line and let students reorder line 3- line 7). Need to test the effect of reordering lines of this code. May not have so significant difference (Maybe some variable simply lost)

The question having recursion is not a same question as the previous manipulating question. Taking the sifting down as an example, the code is a more general method, and it can be used to sift down every tree node. In introductory programming, it is not so common. The introductory programming is more specific and student can easily understand the meaning of each step. However, when the code used to describe higher-level summary of every recursion, it is more difficult for students to understand the meaning of each code fragment. Thus, for this problem, we cannot just cut it directly as we do in the introductory programming. We need to balance the degree of difficulty, which means that we need to give students more hints. For example, we can cut the code into a larger block keeping some structure like if condition or loop. Besides, we can provide some comments for each line, so that students can have some ideas about which code should be put there. The most interesting way to reduce the level of difficulty is to provide the whole ordered code except some key code fragments, and let students insert the code fragments into proper position with proper indentation. For example, we can consider the recursion code fragments as the key code fragments and let students insert them. Besides, we can provide some incorrect distractors with wrong root node (left child node or right child node) or some wrong parameters of next recursion like i, x in sift\_down method. No need to show every recursion in code?

**For database design:**

It is difficult to analysis directly on students’ solution.

For feedback, we should not use a single feedback entity to describe two feedback (specific feedback for students solution, it can be some running result. Another one is the prepared feedback set during inputting question). At least, we need to separate them. And the feedback set during inputting question is more connected with our key issue since it is connected with Parsons problem. But the other type of feedback is more about the studying environment which is not a core point in our project so that we can consider it after we finish the major points.

Report Date: 29/09/2022

**Questions:**

No need to show every recursion in code?

Gantt Chart ok?

Database design ok?

Dynamic Design in chapter 3? Yes, in the final report, it is needed.

**Meeting record:**

Final year project is different from Information System Implementation. In the course Information System Implementation, everything is already defined. The only thing needs to do is to implement it. However, in the course final year project, we do not have prepared clear analysis of the problem. Thus, we need to do further for literal review and analysis for our new idea to show the new idea own use before implementation (since things will be not easy to change after beginning implementation). To be more specific, we can list some typical data structure questions from some textbooks. Summary the characteristic of these questions. Use the previous work in these questions and find the drawback of using this kind of way in data structures and algorithms problem (reflect the difference in introductory programming course with data structures and algorithms course and this comparing can be used as the leading paragraph for detailed analysis). Design a new way to use Parsons problem, that is the usability of our new idea. To avoid reading textbook with no purpose, we need to learn more about existing work to know the gap.

For class and method question, the position of methods can be changed without any effect for the whole question. Reduce the usability of previous Parsons problem.

## Week 7

Report Date: 6/10/2022

UI should look like a game. Overcoming every step looks like beating a boss.

Feedback needs to add icon making it more interesting.

Record students ongoing record, so that students can quit at middle.

## Week 8

Report Date: 9/10/2022

[A document for draft design of using Parsons problem in Data Structures and Algorithms courses (In Chinese) written by me.](https://docs.google.com/document/d/1GksLmOULvJ0YBU-KkVs1qhxdqowqkUmU/edit?usp=sharing&ouid=115115120189815946096&rtpof=true&sd=true)

**Meeting record:**

**Different difficulty levels:**

Different difficulty levels: we can allow students to choose different difficulty levels for hints. For example, we can set three different difficulty levels for coding a class: Level 0, no hints, just cut code in lines and allow students to reorder. Level 1, fix the method names for hints, but the other codes still be mixed, and they need to be reordered. Level 2, fix the method names, and fix the codes for the specific methods. Thus, the codes are grouped by methods, and students only need to reorder the codes in one method instead of several methods. The difficulty levels can be initialized. For some difficult questions like tree, we can set Level 2 at beginning.

**Fix some parts and then let students reorder some smaller parts instead of the whole parts to reduce the difficulty**

We can allow students only reorder some parts of the codes. This part can be some key parts like the next recursion or or just one method of data structures (teach the students one method by one method, not make them feel stressed to face a lot of methods).

**Algorithms analysis:**

We can use comments to introduce the algorithms analysis. For example, we can let students to drag and drop “““O(n)””” to the content of corresponding methods. And also, we can provide several big O and the reasons getting big O (the proof steps to get specific big O)for students to choose the right one.

**Comment:**

We also can let students drag and drop the comment blocks beside code blocks to test whether they understand the meaning of some methods. For example, we can provide the ADT of some specific data structures as comment to let students to insert into the corresponding methods. And the algorithms analysis also can be used as comment blocks. And we can test some understandings of some parts of methods. For example, when should we test whether the stack or queue is empty? Why should test is\_empty when pop() or top() function. This is more about larger block understanding instead of smaller line understanding.

**Tree:**

It is not easy for students to get concrete tree implementation from the abstract one. We can give some concrete one to let students summarize the abstract ones. (But this abstract tree class may not be achieved as well as we think, need more thinking)

**Recursion:**

We can give the specific recursion turns (n=1, recursion first; n=2, recursion two;) to encourage students to summarize the general functions (need more thinking).

**Comparison of stack based on list and stack based on linked list:**

We can give a complete code for one version (for example, stack based on list), and allow students to replace some parts to change it to another version (stack based on linked list). We can fix the method names and let students change the remaining parts. And we also can add some dependencies (for example, when students change the attributes in initiate method, we can highlight the remaining parts using these attributes)

**Other hints about the final report:**

The **dynamic diagram** is needed in the final report.

We can also introduce the **experiment** method in design parts. The implementation is just a middle part or a tool to evaluate the value. We can invite the Year 1 students to test it to know whether it is helpful for students’ study.

Report Date: 13/10/2022

Parsons problem writing style & Data Structures and Algorithms writing style

Stack vs queue compare, heap, binary tree, avl, binary search tree compare

**Meeting Record**

**For report:**

The material can be used for chapter 1, chapter 2, and chapter 3. To be more specific, the chapter 1 and chapter 2 describe the previous work (chapter 1 is in general, and chapter 2 is in detail). All new ideas will be put in chapter 3 (including the limitation, limitations in detail, new ideas, and the advantages of new methods).

Do not use I, me, my in the report.

Read the 2D Parsons problem to see whether the writer wrote something like the feeling of using Parsons problem directly in Data Structures and Algorithms courses.

‘Data Structures and Algorithms’ needs to write like this. And we can introduce the name CS2 when we first show this term.

Parsons problem, Parsons’ problem, Parsons Problem, the Parsons problem all are correct only if consistent.

The new difference between introductory programming courses with CS2 courses is the code in CS2 is highly coupled and related to each other. Introductory programming courses question is less connected, meaning that not use the result from previous?

## Week 9

Report Date: 19/10/2022

**Materials mind map** 1, 2 (Need to add link)

**New Ideas Lists**

Text, letter

Description automatically generated

**Progress Report: To do list (Due Date: 10/11/2022):**

Chapter 1: Introduction

* The objectives need to be checked and updated according to the latest changes and SMART standards**. (Due Date: 30/10/2022)**
* According to the material, establish the relevancy of the problem, and put it in the context of related works. **(Due Date: 03/11/2022)**
* Copy the risks and mitigation from the proposal **(Due Date: 30/10/2022)**

Chapter 2: Background

* According to the material, provide sufficient background knowledge that helps readers not familiar with the problem domain, provide more detail of related works, but do the critical analysis of existing works in Chapter 1. **(Due Date: 03/11/2022)**

Chapter 3: Completed work

* Describe what you have accomplished in the 1st semester **(Due Date: 10/11/2022)**
* Provide detailed analysis of the problem, a high-level design of the system (Wait for writing new ideas, database design, and dynamic design, then done) **(Due Date: 27/10/2022)**
* Highlight difficulties encountered, alternatives evaluated and solutions adopted. (Need to finish it when implementation) **(Due Date: 10/11/2022)**

Chapter 4: Future work

* Describe partially completed work. **(Due Date: 10/11/2022)**
* Project Planning for the 2nd semester **(Due Date: 10/11/2022)**
* Gantt chart **(Due Date: 10/11/2022)**
* Clear idea of what to do to complete the project **(Due Date: 10/11/2022)**

Chapter 5: Conclusion

* Reflect on the progress of the project **(Due Date: 10/11/2022)**

# Second semester W1-W10

You can add sections for W11 and afterward if there are still project development notes after W10.

## Week 1

Report Date: dd/mm/yyyy

## Week 2

Report Date: dd/mm/yyyy

## Week 3 (repeat up to Week 10)

Report Date: dd/mm/yyyy