FINAL YEAR PROJECT GUIDELINES

NOTE
This document is evolving. It will be subjected to revisions with a view to help the current and future students learn and enjoy more from their FYP experience. If you think of relevant points, welcome to let me know and I shall add in your observations.

Revised: 27th March, 2023
Contributors: Dr. S.Y. Yuen and students

The department has a fyp guideline which includes important and comprehensive information. It is in the StudentInfoNet. Please read it carefully. The present guideline gives you some additional advices.

Table of Contents

GOALS

1. To learn to handle a large technical project.
2. To learn good project management skills.

THE PROCESS OF DOING THE PROJECT

1. Identify the project objectives:
   a) There may be one or more project objectives.
   b) The objective(s) should be precise, clear, achievable, and well defined.
   c) The list of objectives should be presented in point form.
2. Write a clear technical specification.
3. Identify the equipment and critical resources needed. Decide on the programming language(s) you would use.
4. Identify milestones. Milestones should be clear, concrete, demonstrable achievements.
5. Prepare a project planning schedule.

6. Revise items 1-5 as the project progresses.

LITERATURE SURVEY

1. To understand what the project is about, you should do a literature survey of related ideas and methods. Read "Useful Database and Web Resources for Searching Information" in my web page.
2. This is the literature survey section in your fyp report. The survey should be written like the introduction of a journal paper.
3. Your survey should comment on existing works related to your problem. For each approach, do the following as best as you can:
   a) describe briefly how the approach works;
   b) describe the application of the approach, report numerical figures, e.g. accuracy, speed, etc. as provided in the paper;
   c) comment on the advantages of the approach;
   d) comment on the limitations of the approach.
4. When deciding on the method to be used. Consider the following criteria:
   a) Is it a recent method? The more recent the better.
   b) How accurate is it? Higher the accuracy the better.
   c) How fast is it? Faster the better.
   d) Is it beyond your ability to implement it successfully? Choose a method that you are confident you can finish.
   e) Are there novel ideas you generate such that you can further investigate and expand on?

PLANNING CHART

1. The planning chart should identify the milestones and assign the estimated date for achieving them.
2. A milestone is a concrete event that one can use to demonstrate progress. A common mistake in the FYP planning chart is that the milestones are vague and one cannot tell whether things have been done or not after reaching the deadline (e.g. study a method). **You should identify and write down concrete milestones** (e.g. implement and debug adding sound recording to an app).
3. The SMART principle used in sports (Specific, Measurable, Achievable, Relevant, Time-bound) can be used to define milestones.
4. Milestones may also be defined following the waterfall model.
5. It is a good idea to break down the project into many milestones. By doing so, one can get a good idea of what needs to be done, and whether the project is feasible. A project is infeasible if some milestones are found to be vague or un-achievable. Also, by achieving milestones in turn, one gets the feeling that more and more are accomplished, and boasts confidence in completing the project. There are many ways to solve problems. Breaking down into milestones is a technique known as decomposition.
6. Some makeup time should be allowed in your planning. Project may and often delay.
7. In a demonstration, you are expected to demonstrate to me something concrete that has been achieved. For example, if you have learnt objective C from week 1 to 12, that is not something concrete and will receive no mark. However, writing a

simple program in objective C that implements part of the project is concrete progress.

TIPS ON PROGRAMMING

1. Form the good habit of commenting your programs as you write them.
2. Read the document "How to write good programs: Structured Programming" in my web page.
3. For each function, under the function name, include a comment section that has the following:

    a) description of the function
    b) description of the input variables
    c) description of the output variables
    d) example of the use of the function
    e) version: this includes the date at which the function is last modified
    f) known bugs

USING SOURCE CODE OF OTHERS

It is permitted to use other people's source code. However, in all occasions (e.g. report, demo etc.), **you must acknowledge the use**. Moreover, you should make sure that the source code is correct. If it is not, your whole system will not work, with wrong and unpredictable results. This would of course ruin your whole project.

One solution may be to use the source code as reference and write your own; another is to thoroughly understand and verify the source code before you make it your own, with the above acknowledgment.

FOLLOWING PROGRAMS WRITTEN BY OTHERS

Stage 1: Get to know the program

1. Backup the original source code.
2. Try to compile and run the code.
3. Find the main function of the program, trace the flow and briefly record the structure of the program.

Stage 2: Understand the code

1. If there is any class or structure defined, look at them first.
2. Record down the meanings of parameters. For example, x= 0: disable, =1 :enable,
3. In C, those structure or parameters are always defined in header file (*.h files).
4. See the function one by one. It is better to understand the basic subroutine functions first (i.e., use a bottom up approach).
5. Add comments after you understand the code.
6. Give a brief description of each function. For example, the input, process, output.

Stage 3: Modify the code

1. When you need to modify the whole function, make a copy of the original function as backup.
2. Add remarks on the lines you have changed. For example, when you modify, for what reason, and what is the original code.
3. It is better to test the function once when you finish modifying every small part, as it is very hard to debug when many parts of the function have been modified.
4. In debugging, it is good idea to check the intermediate states at a given point (for example parameters, flags …); you will get better understand what is going on in the program.

DEBUGGING

If you find that your program has bugs, you can:

1. Compare your current code with the previous backed up one to find the differences made. This can minimize the area to find out the bug.
2. Use a set of synthetic data to calculate the intermediate results for every stage. Use breakpoints to check if there is any difference with the results that you think should be correct to locate the bug.

BACKUP

Always backup your work after each modification. Murphy's Law states that anything that can go wrong, will, and at the time when you most needed it not to go wrong. So beware. Two real experiences: The hard disk of an MPhil student failed when he was writing his thesis! Similarly, the hard disk of an undergraduate student crashed just before his final year project presentation, and he has no backup!

WRITING FYP ABSTRACT

When writing fyp abstract, this is one possible structure:

1. State the objective and significance of the work.
2. Give a succinct summary of your methodology.
3. Give a summary of your experimental findings.
4. Give a conclusion based on your findings.
5. Give recommendations (optional)

Example 1

T. Jalbert , M. Jalbert , K. Hayashi , "State rankings of cost of living adjusted faculty compensation", *Accounting and Taxation,* vol. 1, no. 1, pp.121-137, 2009.

Abstract:
"In this paper we rank states based on higher education faculty compensation. Data on 574 universities across each of the 50 states and the District of Columbia are aggregated to develop a state compensation average. The analysis examines states both on a raw basis and on a cost of living adjusted basis. Rankings are reported for various academic classifications of faculty. Rankings based on salary data alone and

salary and benefit combined data are presented. The results indicate that rankings of states based on raw and cost of living adjusted data are markedly different. The results suggest that faculty seeking employment opportunities should carefully consider cost of living issues. Administrators should design salary packages that reflect the local cost of living conditions in their area to attract quality faculty."

Structure:

1. "In this paper we rank states based on higher education faculty compensation."

2. "Data on 574 universities across each of the 50 states and the District of Columbia are aggregated to develop a state compensation average. The analysis examines states both on a raw basis and on a cost of living adjusted basis. Rankings are reported for various academic classifications of faculty. Rankings based on salary data alone and salary and benefit combined data are presented."

3. "The results indicate that rankings of states based on raw and cost of living adjusted data are markedly different."

4. "The results suggest that faculty seeking employment opportunities should carefully consider cost of living issues."

5. "Administrators should design salary packages that reflect the local cost of living conditions in their area to attract quality faculty."

Variations are of course possible. This is another example:

Example 2:

S.Y. Yuen, C.K. Chow, "A Genetic algorithm that adaptively mutates and never revisits," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 2, pp. 454-472, Apr. 2009.

Abstract:
"A novel genetic algorithm is reported that is non-revisiting: It remembers every position that it has searched before. An archive is used to store all the solutions that have been explored before. Different from other memory schemes in the literature, a novel binary space partitioning tree archive design is advocated. Not only is the design an efficient method to check for revisits, if any, it in itself constitutes a novel adaptive mutation operator that has no parameter. To demonstrate the power of the method, the algorithm is evaluated using 19 famous benchmark functions. The results are as follows: a) Though it only uses finite resolution grids, when compared with a canonical genetic algorithm, a generic real-coded genetic algorithm, a canonical genetic algorithm with simple diversity mechanism, and three particle swarm optimization algorithms, it shows a significant improvement. b) The new algorithm also shows superior performance compared to Covariance Matrix Adaptation Evolution Strategy (CMA-ES), a state of the art method for adaptive mutation. c) It can work with problems with large search spaces with dimensions as high as 40. d) The corresponding CPU overhead of the binary space partitioning tree design is insignificant for applications with expensive or time consuming fitness evaluations,

and for such applications, the memory usage due to the archive is acceptable. e) Though the adaptive mutation is parameter-less, it shows and maintains a stable good performance. However, for other algorithms we compare, the performance is highly dependent on suitable parameter settings."

Structure:

1. "A novel genetic algorithm is reported that is non-revisiting: It remembers every position that it has searched before."

2. "An archive is used to store all the solutions that have been explored before. Different from other memory schemes in the literature, a novel binary space partitioning tree archive design is advocated. Not only is the design an efficient method to check for revisits, if any, it in itself constitutes a novel adaptive mutation operator that has no parameter."

3, 4. "To demonstrate the power of the method, the algorithm is evaluated using 19 famous benchmark functions. The results are as follows: a) Though it only uses finite resolution grids, when compared with a canonical genetic algorithm, a generic real-coded genetic algorithm, a canonical genetic algorithm with simple diversity mechanism, and three particle swarm optimization algorithms, it shows a significant improvement. b) The new algorithm also shows superior performance compared to Covariance Matrix Adaptation Evolution Strategy (CMA-ES), a state of the art method for adaptive mutation. c) It can work with problems with large search spaces with dimensions as high as 40. d) The corresponding CPU overhead of the binary space partitioning tree design is insignificant for applications with expensive or time consuming fitness evaluations, and for such applications, the memory usage due to the archive is acceptable. e) Though the adaptive mutation is parameter-less, it shows and maintains a stable good performance. However, for other algorithms we compare, the performance is highly dependent on suitable parameter settings."

TIPS ON ENGLISH

1. Read "The Elements of Style" by William Strunk and E.B. White. This is a good book if you wish to write good English.
2. Use MS Word spell and grammar check before handing in any document.
3. Use MS Word Thesaurus to find the synonyms of a word. This is useful for learning new usages and alternative descriptions.
4. Have a good English-Chinese dictionary.
5. Form a habit of reading English books you like.
6. The departmental guideline is that reports with bad English will not receive A grade.

TIPS ON PROJECT PRESENTATION

General

1. Different people have different presentation style and ways of organizing a talk. The following are suggestions only. You may do it in your own way as long as you present it well.

2. Have a good model of your audience. One very common mistake is to assume that your audience is as familiar as you about the background. Usually they are not. Assume your audience is intelligent but do not have the background. It is important to explain the background and motivation clearly.

3. Prepare your slides with the level of audience in mind. For example, a very **different talk** should be prepared if you were speaking to secondary school students, university students, or colleagues.

4. Your presentation should be self-contained. No symbols should be left unexplained. You should introduce a concept the first time it appears.

5. You are probably the person who is most interested in the topic you are going to present. Do not expect the audience to be as interested as you; instead, it is your job to make your presentation enthralling.

6. Maintain eye contact with the audience.

7. Face the audience. Do not show your back to the audience at any time. It may happen when you are explaining or reading the slides from a large projector screen.

8. Do not read from the slides. You need to be sufficiently familiar with the slides to be able not to constantly refer to them. Thus, you need to read the slides and think about how to present them before the presentation.

<u>Preparing your presentation</u>

1. In your first slide, show the title of your project, your name, and the name of your supervisor and assessor.
2. In your second slide, show a list of objectives. Make sure that they are clear, in point form and well defined.
3. Some slides on the motivation and background of the project.
   Motivation : *Why* the project is important / worth doing.
   Background : What past works by others have been done on the problem.
   Do not have too many slides on the background, as you would not have enough time to talk about your work.
4. The bulk of the presentation is what you have done in the project. Divide it into
   a) Theory /Methodology:
      - *How* it is done. The assessment panel consists of professors who may not be working in the field. You must present your work i) technically of a high standard; ii) clearly understood by intelligent non-specialists.

   b) Experimental Results :
      - *What* has been done.

- Include the full experimental setup, e.g. what machine, language, control parameters, threshold settings, …
- Data - Organize your data into tables, graphs, images.
- Make use of animation in power point to present and highlight your data and findings.
- If possible, include a demo, which would make a more attractive presentation. There are two ways to prepare a demo.
  i) *live demo* - This will gain many marks if the demo works, but Murphy's law states that if a thing can go wrong, it would. Therefore, if you decide to do a live demo, make sure that it would work. Also prepare a recording as backup. If you have hardware, bring the hardware and show it to the panel even if you are not going to live demo with it.
  ii) *recording* - This is a safe alternative to doing a live demo. You record a live demo and play back. Make sure that it plays back. Test it beforehand.

5. Use a slide to conclude. Your conclusions should be in point form. Each point should clearly state what you have achieved or found.
   It is a good idea to think about the question: "Why should we give the project an A?" and try to answer this question convincingly in your conclusion.
6. The last slide is a slide "Q and A".
7. If you do not have enough time, it is a good idea to prepare additional ppt which provides more detailed information. You may think of possible questions and prepare ppt that answers them. These ppt needs not be shown. However, if the exact question is asked, show the ppt. It gives an impression that you are well prepared.
8. Check the grammar of your ppt. Make sure that the English is good and there is no spelling mistake. Use grammar checking tools.
9. Rehearse your presentation beforehand. It is a good idea to rehearse it with fellow fyp students. Ask them to jot down comments and observations while listening to your presentation and give you feedback for improvements afterwards. It is also a good idea to listen to good presentations by fellow students and check out how a presentation is run beforehand.
10. There is an assessment form used by panel members. Find out the assessment criteria and the rubic used.

<u>The Presentation</u>

1. You are provided with a laser pointer. Familiarize yourself with how to use it beforehand.
2. Load your software into the PC before the beginning of the session.
3. After the last presenter finishes, you should get your ppt running immediately (e.g. put it on the desktop.)
4. Be calm and composed during your presentation. Speak with confidence.
5. Always try to maintain eye contact with the audience.
6. Stand upright with your back straight.
7. Formal dressing is recommended. Marks will be deducted if you appear unprofessional.
8. Keep to the time. Do not over-run. Rehearse beforehand and check the time. You may also video your rehearsal to see how well you are doing.

9. Speak fluent English. If your English is weak, rehearse more.
10. Some students may like to bring a deck of small cards. I do not recommend it. If you must do so, only jot down the main points; and maintain eye contact. Personally, I prefer to use the main points on the ppt and elaborate rather than use cards. Better still, become very familiar with your lecture so that you do not need to refer to the ppt.
11. Enjoy the presentation. Treat it as a valuable learning experience.

Q and A

1. Listen carefully. Make sure you understand what the question is before answering. The assessors find it annoying if you are not answering their question.
2. If really in doubt, put the question in your own words and ask them whether it is really the question they are asking.
3. When faced with a difficult question, you can buy a little time by saying "This is a good question", but you have to give them a good answer right after that.
4. If you really cannot answer the question, say so. People find it annoying if you pretend to answer the question, but actually does not.
5. Be polite. Have good etiquette.

TIPS ON DEMONSTRATION

Respect Murphy's law. Prepare well. Unexpected things may happen, especially for hardware projects. Thus, it is important to have a good preparation and a backup plan.

Before the demonstration, arrive much earlier to the laboratory. Do not let your supervisor /assessor wait while you set up.

Set everything up and make sure that everything is running properly. Check and re-check. If you have time, it is better to prepare a video as backup, just in case the demo cannot be done for some unexpected reasons. Then you can show the video and ask for a second chance to sort out the unexpected problem.

FORMAT FOR QUOTING REFERENCES

Check that all references follow the same, consistent format. It will look unprofessional if otherwise.

When quoting references, make sure that all details are correct. Check it over two times. Incorrect references are unprofessional will give a lot of troubles to the readers who want to look up the references.

The Google Scholar has a reference format that can be directly imported into Latex.

For the IEEE format:

Conference Paper:

[1] T. Bäck, "The interaction of mutation rate, selection and self-adaptation within a genetic algorithm," in *Parallel Problem Solving from Nature*, R. Manner and B.

Manderick, Eds. Amsterdam, The Netherlands: North-Holland, 1992, vol. 2, pp. 85-94.
[2] C.W. Sung, S.Y. Yuen, "On the analysis of the (1+1) evolutionary algorithm with short-term memory," in *Proc. IEEE Congress on Evolutionary Computation* (CEC), June 2008, pp. 235-241.
[3] T. Friedrich, N. Hebbinghaus and F. Neumann, "Rigorous analyses of simple diversity mechanisms," in *Proc. Genetic and Evolutionary Computation Conf.* (GECCO), July 2007, pp. 1219-1225.

Paper within an edited book:

[4] M. Kimura, "Overdevelopment of the synthetic theory and the proposal of the neutral theory," in *The Neutral Theory of Molecular Evolution*. Cambridge, MA: Cambridge University Press, 1983, ch. 2, pp. 25-33.

Journal Paper:

[5] G. Rudolph, "Convergence analysis of canonical genetic algorithms," *IEEE Trans. Neural Netw.*, vol. 5, no. 1, pp. 96-101, 1994.
[6] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Trans. on Evolutionary Computation*, vol. 1, no. 1, pp. 67-82, 1997.
[7] A.E. Eiben, R. Hinterding, and Z. Michalewicz, "Parameter control in evolutionary algorithms", *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 2, pp. 124-141, 1999.

Book:

[8] J.H. Holland, *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI: Univ. Michigan Press, 1975.
[9] F. Glover and M. Laguna, *Tabu search*, Kluwer Academic Publishers, 1997.

PLAGIARISM

1. Everything you write must be original. No cut and paste is accepted, even if you acknowledge the source.
2. The definition is: No two sentences should be the same. It is ok if unintentionally one sentence is the same. It is usually unacceptable when two or more consecutive sentences are the same, though of course it should be judged case by case.
3. If a figure is not drawn by yourself, you should acknowledge the source in the legend. [When you publish a book, you need to ask the author of the figure for permission to reprint the figure in your book.]
4. Before your report is submitted, you need to submit it to an anti-plagiarism system for plagiarism check. Plagiarism will lead to failure or lowering of grades.
5. If a survey is done jointly by two students, then in the report, you should acknowledge at the beginning of the survey that it is written by two students jointly.