THE HITCHHIKER GUIDE TO IT PROGRAM DESIGN

This document will serve you as a guide to working all semester in the IT Program Design offering. So without further ado, jump in and welcome to COP 3515 IT Program Design!

"You are here"

Let's start by a bit of orientation. This course is an online asynchronous offering. This means that we do not have set times for face-to-face meetings or for online sessions to which attendance is mandatory. This doesn't mean you are expected to spend less time working on this course though ☺.

The canvas site we will be using this semester will be our hub to interact with one another. We will be using discussion forums to address questions you might have, emails for anything personal, but also web-based meetings via the Blackboard Collaborate software, aka Elluminate. I strongly encourage you to book some time one-on-one with the instructor and/or the TA. We can accommodate you for meetings on campus or, if you prefer, online with the above-mentioned tools freely available from the course site itself.

These tools are really suited for teaching programming; they allow us to share desktops with students and show them how to work through some of the exercises. This software is easy to use and freely available; simply click on the "Blackboard Collaborate" link, find the "student lounge" and experiment in the virtual room. You may also email your instructor or TA to have them help you get started with this tool. We will also be happy to facilitate online study group meetings by providing you with all the necessary logistics for them to happen. Just email!

Overview of the course's site

There are really two different sites which provide information on this offering. Both sites refer to the "modules" we will be working on. While it's natural to want to "peek ahead", we really recommend that you follow closely the announcements on the LMS site which will tell you exactly what you need to work on every single week of the semester. It's never pleasant to realize that you missed deadlines, and most likely points, just because you skipped ahead without paying attention to the announcements.

The Website

It is located at http://cereal.forest.usf.edu/clue/progdesign/. You will find a link to it in the "modules" section of our LMS site for convenience but it's not a bad idea to bookmark the above URL.

You will find there all the "static" material we are using; videos, PDF, assignments descriptions…

The LMS site – Learning Management System

This is where you are right now if you were able to download this. We will use the LMS to send emails, discuss on forums, submit exams, take quizzes, meet online…

Overview of the whole semester

As you already know, our first week is being devoted to "getting started" while late enrollees join us. If you are taking this offering during the summer, the first week also deals with the first part of module 101. Refer to your instructor announcements to figure out exactly how much you are expected to do during this week.

The final week will also be "special" in so far that it is only meant for final exams.

However, the rest of the semester will be organized into 2 week long modules. We are going to start you off by taking a look at the topics we'll be covering in these modules. They roughly fall into 3 parts, each with their own specific learning goals.

Part #1 – Reviewing & Transitioning to the new language

The first two modules, [101] and [102], are mostly review modules in so far that all of the concepts introduced in these have been already covered in the IT Programming Concepts Pre-requisite. The only new thing here is the specific syntax of the C language which is close enough to Java or C++ which you have already studied.

Make sure you use these modules to review what you might have forgotten since the last time you took IT Programming Concepts. If you experience difficulties in these modules, your pre-requisites probably forgot to teach you how to program. Email directly the instructor to see how the situation might be salvaged.

Please note that some "Refresher Material" has been also made available on the Canvas site. This section of our Canvas site holds modules which are used during the first 3 weeks of IT Programming Concepts. It is worth taking a look to help you assess whether you're ready for this offering, especially if modules [101] and [102] are hard on you. If you are unable to understand the lectures or do the exercises on your own without watching the solution videos in these "refresher modules", email your instructor immediately.

Part #2 – Technical Aspects of the language

Modules [201], [202] and [203] represent the most technical part of this offering. We add new programming concepts to the previously reviewed ones and progressively start looking deeper into how things work at a lower level than what you were exposed to in IT Programming Concepts.

The objective is to strengthen your programming skills by allowing you to understand the "rules" you've been so far following pretty blindly; e.g.

- What is passing a parameter by reference or value all about?

- How are variables stored in memory?
- …

This part is also the part which will allow you to gain this "under the hood" understanding of how things work which you will later be able to leverage every time you learn a new language or have to troubleshoot a program.

Part #3 – To Data Structures and beyond!

This part is generally skipped during summer terms.

The last modules of this offering, [301] and [302], are meant to allow you to review everything we introduced so far and apply it again to a different purpose.

You will have an opportunity to reuse pointers and recursion in a totally different situation in order to gain some more practice with these difficult topics but also experience hands-on one of their main applications.

We also introduce how to build data structures in your programs, which is going to prepare you to take IT Data Structures successfully.

How to work on each Module

Each module in this offering will be 2 weeks long. New material is released every Monday at 11:55PM and assignments in it are due by the following Monday 11:55PM. This routine is stable throughout the semester to ensure you get "in the groove" with this offering within 1-2 week.

Each week has a specific role in your learning process and all activities for it must be completed before the next Monday deadline. You need to start working through the steps below as soon as the material is released and do it over as many days as you are able to in order to provide opportunities for discussions / exchanges with your peers, TA and instructor.

Let's take a look at what you'll have to do during each week of a module.

Module's first week

The first week is used to lay the foundations to your knowledge. It involves mostly passive learning where you are acquiring new facts, new procedures. Expect a lot of *memorizing* and *understanding* being your learner focus during this first week.

The active learning part for this week is mainly in your interaction with the so-called "Peer Learning Forums" – PLF – and the "apprenticeship exercises". These will respectively allow you to flex your ability to *synthetize* information in order to ask helpful questions, and your ability to *apply* what you learned.

It is also an opportunity to use the module's Practice Quiz – PQ – to help you prepare for its graded counterpart; the Graded Quiz – GQ.

Overall Idea – Learning by Reading and Asking questions

In most face-to-face offering, an instructor uses slides to lecture about a topic. This is not sufficient to teach programming and serves only to introduce students to the minimum knowledge for them to start really learning hands-on with exercises.

In this offering, this first exposure to the necessary knowledge is achieved by reading. We then engage in various activities which strengthen the learning. Online offerings, when done right, make both the students and educators work explicitly of developing learning rather than just "Listening to the smart guy on the stage". However, this takes some more efforts and not everyone knows exactly what to do in order to reap the benefits so let's discuss this a bit. Feel free to email your instructor anytime if you have questions about the learning process itself. It's important you know exactly why you are being asked to do specific tasks to support your learning.

You will also find that developing the skill to learn from written documentation is something which is critical for any IT professional. By learning the way we do, you'll have an opportunity to start developing this skill now, while you still have an instructor available to help you out as needed.

The following sub-sections will provide you advice on how to work on your reading assignments while also using the Peer Learning Forums – PLF – to get help with what's causing you problems.

First reading – mapping out the module

As soon as the module is released, proceed to read the chapters assigned by the **Reading Assignments** and take a look at the **Module Overview** PDF. This first reading is meant to just help you figure out what the chapter is all about. You should be able to get it done in an hour or so, depending on the amount of reading assigned, then proceed with the other steps another day. While you read, write down;

- The new factual knowledge introduced; this is the stuff you need to memorize: how something works "under the hood", new syntactical rules, good programming tips, warnings about usual errors…
- The new procedural knowledge introduced; this is the "how to" part of things: how to write a program from scratch, how to use recursion to solve problems, examples of programs…
- How any of this relates to things you've learned in your previous programming offerings. Learning by analogy to what you already know is important.
- Any questions which come to mind while reading should be noted somewhere for you to refer back to them during your next reading.

During this first step, the **Peer Learning Forum** for the module you are working on will not really be useful; you need to first do an in-depth reading before to have good and useful questions to ask.

Second reading – learning and asking

This time, you are going to read the chapter more slowly and much more carefully than the first time. Read the chapter in depth and, as you do so, keep an eye on whether you are actually learning both the factual and procedural knowledge it conveys;

- Factual knowledge is the easiest to absorb, just memorize it. Take the various factual knowledge bits you identified at your first reading and try to explain them to your pet. If you're unable to articulate the thoughts or some things are not clear, you have a great question to post on the forum:

  "This is how I understand recursion, but I am not seeing why the textbook makes a difference between a recursive call and a recursive definition. Am I missing something?"

- The procedural knowledge is harder to absorb, it requires a bit more cognitive involvement, that's why. If you just read about top-down stepwise refinement as a way to break down a problem, you need to verify that you understood the idea and are able to repeat it to someone else. But you need also to make sure you are able to use it! Take an example and try, if you're not able to do it, you now have a question for the forums!

  "I tried to apply top-down stepwise refinement to the following problem I made up. As you can see below, I'm not sure how to move from step #3 to step#3. Do I need to get to a point where I have code??? Is what I have down enough?"

Doing all of this is making your reading through the chapter slower, isn't it? Good. This means you're actually learning something. What about these questions you wrote down during your first reading? Some of them should be obvious now, we all miss stuff the first time we read. Others are going to be still just as cryptic, we're going to focus on these at our next step. Others will just transform a bit now that we have been going a bit deeper into the material, that's actually very good.

PQ – Practice Quizzes

Every module has a PQ which allows you to practice your understanding of the reading assignment. It's a good idea to use these in parallel with your readings and post questions on the PLF on anything you do not understand.

These are also a great preparation for the module's graded quiz.

PLF – Posting your questions on the module's Peer Learning Forum

As you are reading, or watching videos, make sure you push yourself to do it right. This means that you are going to have to learn to assess when you do not understand something.

- Some people can read an entire textbook on quantum physics in 10 hours and "feel" that they get it. Then, they might be unable to solve any physics problem. This means that they are too optimistic about their capabilities or have not thought that they ought to put their knowledge to the test to make sure it's solid.

- In programming, this is similar to someone writing 5,000 lines of code and assuming it works. Programmers always think what they write is right. No one writes "bad code" on purpose. Yet when you compile then test your programs, you find that you missed things. Same goes with learning from a lecture, a video or a book.

So you want to be careful about being able to realize when you need help and when you don't. If you feel you have issues with this, you need to email personally the instructor to work on fixing this.

The PLF – **Peer Learning Forums** – are also there to help you with this.

- There is one PLF for each module. The instructor will give you deadlines by which you need to post your own questions in new threads and attempt to answer other students' questions.
- PLFs are there to allow you to ask the questions you identified during your reading to the instructor. They also allow you to engage in discussions with students having similar difficulties, in a manner which helps you overcome your learning barrier.

There are a few things you need to understand about how the PLF will help you learning;

- Your second reading should provide you with plenty of questions to ask.
- You need to craft them so they actually help you learn what is getting you stuck. This often entails taking time to put details in them, explain what you read, how you understood it, add a few lines long program you wrote to test out your understanding, show the "unexpected results" it gave you so others might help you by fixing the program or explaining how it executed…
- Do not simply post this kind of question; *"I don't understand section #3"*. When working with other IT professionals, or asking for help in programming forums on the web, you will be literally shunned if you keep asking for help without showing you made a good effort at understanding. It underserves you as a learner to do so in this offering and other students might feel you are simply attempting to "leech" their time.
- This being said, there are times where you will feel genuinely totally lost on an entire section of chapter. Do not despair, take the very first thing you do not understand and focus your efforts on crafting a question on this instead. There is always a first thing that get your eyes glazing, some students keep reading thinking it will get better but the misunderstandings accumulate into an avalanche which makes you feel like the whole chapter is lost to you. Develop your awareness of the exact moment where you starting losing your footing. This is key to improving your learning skills.
- Here's an example of a question with enough details in it to allow others to understand your problems and come to the rescue;

  "I think I understand this and that, see my definitions below and please correct me if I'm wrong! But then when I read section ## on page 233, it contradicts what I thought I understood from page 369".

- Before you post a new question in a new thread on the module's forum, make sure someone else didn't ask already the same thing! If there is already a thread about a similar question, post yours under it instead of creating a new forum thread. This wis also where you'll find your answers so you might want to *subscribe* to this thread to get an email every time it is updated.
- Last but not least, never hold back questions! You're here to learn, not to impress your instructor or your peers. If someone does not accept your questions as a learning opportunity, they are out of place in this offering, not you.
- Sometimes, by the time you're done explaining your problem in a question, the solution will show itself and you will have one of these Eureka! Moments. This is perfectly normal. Other times, you will post the question the find the solution minutes after you hit "post". Again, perfectly normal and expected in this offering! Other times, you will need someone to respond to your question to get you unstuck. All this is fine, the essence of the PLF is to help you learn ☺

Attempting to answer others' questions on the module's PLF

The PLF has another way to help you out. Remember when we talked about the fact that some students might read the entire chapter and feel like they understood everything until they try to actually do something useful with that knowledge?

Well, PLFs may help you develop the ability to read more carefully and know when you are potentially missing something. To do so, take a look at the questions your peers posted on the module's PLF. If you posted all the questions you had already, then this means everything else in your readings was understood, right? Good, let's see if this is really the case ☺

- If you read another students' question and you do not see the response, then the PLF just helped you identify something you *thought* you understood. Go back to the reading material, looking specifically for information about this one question. Attempt to answer the question, the instructor or TA will let you know whether you're on target or not, thus validating what you just learned
- If, on the other hand, the other student's question seems easy to you, still take the time to share what you think the response would be. The idea here is that you may think you know and even be able to explain what you understood to someone else, yet still be wrong about it! This happens to everyone but it's hard to detect in an online offering until students actually say something. So by trying to help other students, you are really validating your own understanding.

This helps you do a **third reading** of the material but this one is much more problem-driven which tends to make it more interesting and more effective in helping you acquire new knowledge.

Another thing; it does not matter whether you end up being right or wrong, both are opportunities for the instructor and TA to detect a misunderstanding and provide you with guidance on fixing it at the end of the PLF activity when they review all posts.

However, it matters that you put efforts into responding to your peers. This is while you craft your responses that the learning will most likely occur. Again, simply posting "*I*

have the same problem!" or "*beats me too!*" is great to build conviviality in the forum but won't help. So while you are allowed to also just "chat" on the PLF about the problems encountered, your instructor will focus on the responses you post which show a good attempt at helping the other students when estimating your participation.

Apprenticeship Exercises Videos

Apprenticeship exercises are an absolute must-study! So far, most of our learning activities revolved around making sure you absorbed factual and procedural knowledge from your readings. Next week, you'll have to apply all this so we need to start the transition between *absorbing knowledge* and *applying* it.

These exercises will help you establish a bridge between the theory you just learned about and how to apply your knowledge to actually write programs from scratch. Most introductions to programming you already took, do not put you in a position where you have to both design and implement a solution from scratch. Often, the "requirements" do not only specify what the customer wants your program to do, but they also tell you how the program should be written! In this offering, you are responsible for determining *how* your program will meet the requirements. The customer who wrote the requirements might not even know anything about programming!

To help you on this learning path, each apprenticeship exercise is described in a PDF or DOCX file as if it was coming from your client / boss / exam. Read it carefully, these are the *requirements* of the work to be done. Start thinking about how you'd apply what you learned to do the job and open your programming environment on your machine, you're going to need it to actually learn.

When you think you have an idea, start playing the video accompanying the exercise. These videos implement a so-called "cognitive apprenticeship" approach to teaching in so far that they are going to show you, step by step and including errors, how someone who knows how to program would tackle the problem. It is important that you realize that giving you the solution files directly would be much less work for both you and I. Unfortunately, this is not how you teach programming.

The textbook shows you plenty of problems and their complete solutions. Most students just memorize "when I see problem X, I need to produce code Y". Great. Not really. This approach only forgot to teach you one thing: how to get from a list of requirements defining a problem to a solution. This is called programming. Most of you didn't have to walk this entire path from requirements to solutions in previous offerings. The videos are there to not only give you a full working solution but also expose the process of getting there step by step.

Many times, you'll find "mini lectures" embedded in the videos. Make sure you watch them entirely, pause them often and take the time to think about how you'd get to the next step before playing the video a bit more to check if you were right. Most important; ask questions about what you watch on the forums!

Last but not least, you may work on these exercises as early as your second reading and use the PLF to ask questions about anything problematic.

GQ – Graded Quizzes

The GQs are not there to help you learn, they are there to evaluate that you are ready to proceed to the 2nd week of the module.

It's a good idea to save them for last, and spend the module's first week getting help from the PLF about the reading material and the apprenticeship exercises.

These quizzes are generally about one hour long, refer to postings for details. You are allowed to take it only one time. If you get disconnected, reconnect immediately and work on answering the remaining questions. The LMS periodically saves your answers for you so you can immediately resume the quiz where you left off. When you're done, email the instructor to let him know about any issues you had.

If you start the quiz at 7PM and its duration is one hour, you have up to 8PM to submit all the responses. No matter how many times you reconnected. Make sure you have a reliable internet connection and do not wait for the last minute.

Module's second week

Week #2 of each module is there to help you *apply* and *generalize* your recently acquired knowledge and prepare you for upcoming graded assignments.

"Wrap Up"

The Monday following the release of the module, the instructor will focus on making sure every single question on the forums is answered fully. Most of the time, the instructor and TA will keep an eye on the forums all week long and jump in to help. However, the first week is there to allow everyone to brainstorm, post questions and try to answer each other's questions. Regardless, any question which was left for you to discuss among yourselves will be addressed by the following Monday 11:55PM.

The instructor will also take a look at all the attempts at responses and post to indicate whether the student was right or wrong and why. This is why it's important you try to answer other students' questions! Otherwise, you might see an answer from the instructor which will not help identify something you misunderstood. Always think out loud on these forums so that someone might correct you! It's not being right that matters on these forums, it's learning something.

Graded Quiz Solutions

Solutions for the graded quizzes will be released after the Monday 11:55PM deadline. It's a good idea to revisit the quiz, look at your mistakes and email your instructor for help understanding what you did wrong.

PA – Programming Assignments

One programming assignment will be released in each module. PAs are meant to verify that you not only learned *about* programming but that you are actually able to do something with this knowledge! To pass this offering you need to know how to write programs from scratch. No solutions will be provided but you are allowed to

- Ask questions to clarify what is exactly expected using the module's forum
- Exchange tests with your peers, use the tests of other students to be able to verify your program does everything which is expected and does it right. See Peer Testing below.
- Discuss difficulties you are encountering to get some advice on the module's forum. No code should ever be posted or exchanged in these posts, but you're free to discuss the obstacles you're dealing with in plain English.

You should plan on devoting at least 12 hours to actually program a solution to these assignments. You learn programming by programming. It is fairly obvious to state but most students don't do it because it's time consuming. This is an error, needless to say.

All week long the TA & instructor will be available to help on the forums with the practice assignment.

PT – Use Peer Testing to validate your PAs

The same way the PLF module forums help you with your reading assignments and apprenticeship exercises, we have an activity meant to get you help with your PAs.

Most of our PAs, except the very first ones; i.e. PA101 & PA102, will require you to write tests. Your instructor will explain how and when you are allowed to share your tests with other students since the method we will be using will vary between the various PAs. Regardless of the method we use, this will allow you to get a bunch of other tests to apply to your program. Benefits generally include;

- Realizing you didn't implement a required feature when your program fails another student's test. This also allows you to improve your list of tests!
- Realizing you misunderstood a feature. The way the other student tests the feature is different than yours.
- Realizing your implementation has a bug; you implement the feature the test is validating but you do not get the expected result. Your tests should have a new test too obviously

Obviously, you need to be careful. Sometimes, the other student is wrong so you still need to develop your own programming skills to identify good tests. Talking about, or providing parts of, the PA solution in anyway via your test files is not allowed.

NED – Use NED to validate your PAs

As you work on your programming assignments, keep in mind that we have another web tool meant to help you improve the quality of your programs before we grade them. How to use NED? Simple.

If you are using Code::Blocks by itself, you may simply use a web browser instead;

- When you're working on your PAs or GPAs, open a web browser at http://CEReAL.forest.usf.edu/clue/ned/
- Use the **browse** button to select your source files, one by one. You should have to do this only one time after you open the browser.

- Use the **analyze** button every time you want to have these files validated by our server.

When you upload your files, several things happen;

- They are compiled using the GCC compiler – the same you are also using in Code::Block – but with many more options enabled to detect all sort of errors
- They are analyzed using a few state of the art static code analysis tools which will try to detect bug patterns in your work

A few things to understand;

- If your program has errors these tools are unable to understand, then you won't get warnings. This doesn't mean things are perfect
- If your program is perfect, you will not get any warnings either – a good thing
- Sometimes you will get warnings but, after closer inspection, you will realize your source is still good. This is because these tools still produce false positives. Their goal is to warn you about anything suspicious, even if it's most likely alright. Just do not start assuming you "know better" and ignore all warnings.

In addition, some of the warnings produced by the compiler or the static code analysis tools have been hyperlinked to "tutorial pages". We did this for the messages which confused students in the past semesters.

- Many warnings / errors won't have a tutorial; it's simply because students so far have not mentioned having problems understanding them. Use the "request tutorial" button to let us know if we should write a tutorial on a specific message. We just don't want to flood you with tutorials on trivial things; this would just waste your time reading through them. On the other hand, we want to know what we may add to get you more useful tutorials.
- If you read the tutorial but it's not really helping, use the "yes" / "no" buttons at the bottom of the page to give us feedback. We will soon implement a better feedback system but, in the meantime, never hesitate to drop me a line at alessio@usf.edu.

When you provide feedback, keep in mind the following;

- The NED team is not grading you. Do not be afraid of sharing your thoughts about the tool with me, all it will do is make sure I am able to improve things this semester for you & your peers. I just ask that you be specific about what is missing or what you'd like / expected to find.
- When you use NED, everything is anonymous. Well except if you put your name in the source files but even then, I do not share the source files with your instructor.
- While NED keeps statistics on its usage, everything is anonymous unless you upload files with your name. Make sure you read the "terms of usage" on the web page, we do not care about keeping tabs on who did what. We are trying to identify global trends instead to improve the help provided by the software.

The tutorials are also summarized in a table below the file uploads area so you may just peruse through them to learn. As students use NED & provide feedback on the tutorial pages, you'll be able to see in this table the following statistics;

- Views – how many times the tutorial has been read
- Useful – how many times a student voted "yes" on the mini survey for this tutorial
- Useless – same but with "no"
- Freq – how many time the errors related to this tutorial have been detected in the source files uploaded by students

Where to get help?

Last but not least, we have a whole team to support your learning. The following is a list of types of situations, along with the resources available to help you with them;

I have a problem with one of my **GQs** or need help understanding the grade / feedback I received	- Email your instructor. - This is actually the only course of action allowed for graded assignments. See syllabus' academic dishonesty sections
I do not understand something in the **textbook**, **apprenticeship exercises** or **PA**	- Post a question on the module's PLF - If you need a response urgently, still post on the PLF but then email the instructor to let him know you have an urgent question - When you post questions on the PAs, remember that you should not post any code
I need help with my learning process itself	- Email the instructor
I have problems with the "NED" or "WebPT" web tools or I am not sure how they might help me	- Email the instructor
I have a problem with one of the **exams**, or need help understanding the grade / feedback I received	- Email the TA and copy the instructor - This is actually the only course of action allowed for graded assignments. See syllabus' academic dishonesty sections