

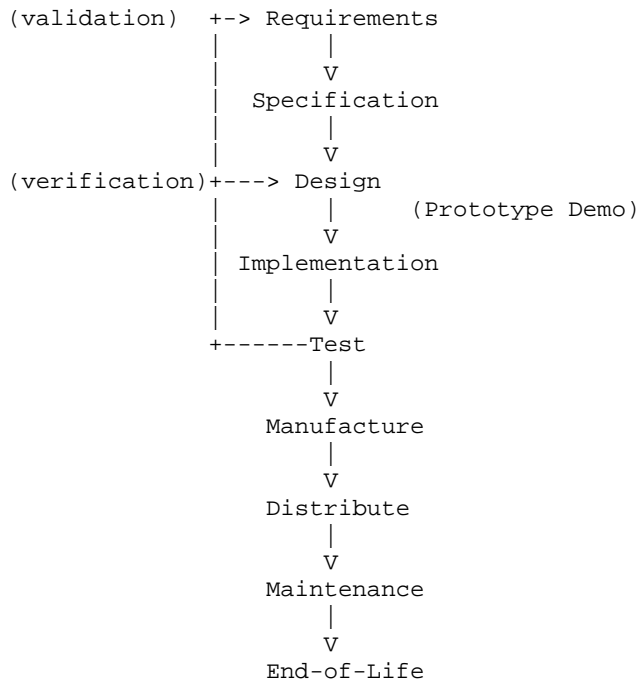
>>> SOLUTIONS <<<

Welcome to your 10% mini-exam for Senior Project. You have 75 minutes for the exam. The exam is “anything on a dead tree is allowed” as a reference. That is, you may have anything with you on paper including copies of old exams, your notes, your book, and so on. You may not share material or give help during the exam. If you copy your answer directly out of the textbook (or some other source), be sure to properly “quote” the answer (otherwise it is assumed that any answer is given in your own words). Read the last sentence again. There are six questions each worth 16 points and one extra credit question worth 10 points. Please use a separate sheet of paper for each question, do not write on the back of any sheet of paper, and submit your exam with this cover sheet as the topmost sheet and the problem sheets following in number order. You get 4 points for just following these instructions correctly.

Use of a laptop for completing the exam: These problems require a lot of writing. If you are like me, you can type better than you can write by hand. So, I will allow you to use your laptop to type the answers to this exam. At the end of 75 minutes I will ask you to hand-in to me your thumb drive with your exam file (name the file with your last name)... we will then go up to my office and print-out your exam (and return the thumb drive to you). It is your responsibility to make sure your file is properly saved and not “eaten by the computer”. During the exam you may only have your word processor open to your exam answers file, and nothing else. You will be asked to turn-off the wireless (WiFi) in your laptop. The exam will be proctored from the back of the room. Anyone opening-up a Web browser or otherwise displaying a file that is not the exam solution text will receive a zero for the exam.

Problem #1

Sketch the product development process as we have discussed and followed it in this class.



1 pt for each listed term, except 2 pts for Requirements, Specification, Design, and Prototype demo.

(yup... you can copy this for pretty much all previous old tests. Just by copying it, maybe you learned it just a little bit better!)

Problem #2

Fill in the blanks. Each blank is one word, or a short phrase (about 2 to 4 words).

1 pt for each completely wrong question, 0.5 pts for partially wrong.

- a) The requirements document addresses **what is the problem**
- b) The items in a requirements document must be **measureable, numbered, and in priority order**
- c) The specification document addresses **how to solve the problem**
- d) A specification will contain precise descriptions of **inputs, outputs, and transformations**
- e) A specification document will always contain a **traceability matrix** (hint: has to do with requirements)
- f) Key words (or phrases) for the formal definition of design include **process, meet desired needs, iterative, convert resources optimally, synthesis, and analysis.**
- g) Standards are an **agreed-upon way to do something** (hint: more than 4 words)
- h) Testing consists of **verification and validation** (hint: both words start with a “v”)
- i) A testcase has four parts, which are: 1) **name**, 2) **system configuration**, 3) **input**, and 4) **output**
- j) Project plans are often shown on a **Gantt** chart where each **milestone** must be measurable.
- k) Too little process may result in **chaos and no productivity** (hint: key points)
- l) Too much process may result in **too much overhead and no productivity** (hint: key points)
- m) Designs reviews have four purposes, they are: 1) **feedback**, 2) **education**, 3) **evaluation**, and 4) **contractual**.
- n) Writing documents forces **gaps** to appear
- o) Documents can be used as a **checklist** of progress
- p) “The task of a manager is to **derive a plan and then realize it**” (Brooks)
- q) In a presentation the rule of thumb is **one slide per minute**
- s) A press release show contain the **5 Ws and 1 H and a user quote.**
- r) A poster is a **standalone** presentation
- t) In a poster all figures should be **captioned and called-out** in the text.

Problem #3

Answer the following questions about the key documents in our development process. Hint: the course grading guidelines cover the answers. Also, parts of the answers are in the solutions to problem #2.

1 pt for each missed attribute.

- a) List five key attributes and/or components of a requirements document as taught in this class.

Here are six attributes and/or components from the grading guidelines for the class:

1. Does the introduction provide sufficient background for understanding the requirements, but not itself describe requirements? Are all non-obvious terms defined in the introduction?

2. Are all things given to you listed in the assumptions? Are the assumptions reasonable and non-trivial?
3. Are the requirement items all numbered and measurable?
4. Are the requirement items in order of priority for the problem being solved?
5. The requirements does not describe implementation, but can be reasonably implemented?
6. The document is formally written, well formatted, without grammatical and typographical errors, contains names, contact information, version, date, and page numbers in the form of page x of y?

b) Repeat (a) for a specification document.

Here are ten attributes and/or components from the grading guidelines for the class:

1. Are the "minor details" including names, contact information, date, version, and page numbering all in order?
2. Is there an introductory paragraph that sets the stage for the document?
3. Are unusual terms defined and specialized concepts explained?
4. Is the syntax described? This might include descriptions of fields and user screens?
5. Are the semantics described? This must include a diagram of some sort (such as a flowchart or FSM)?
6. Are all input, outputs, and data transformation described?
7. Are appropriate standards identified and properly referenced?
8. Is there a traceability matrix (with trace back to requirements)?
9. Could this document be given to a third person - someone who is "skilled in the art" - where this person could correctly implement your project so as to meet your requirements?
10. The document is formally written, well formatted, without grammatical and typographical errors, contains names, contact information, version, date, and page numbers in the form of page x of y?

c) Repeat (a) for a test plan document.

Here are five attributes and/or components from the grading guidelines for the class:

1. Are test cases described so that someone completely unfamiliar with the project could execute them?
2. Do all test cases descriptions include a) test case name, b) system configuration, c) exact input, and d) exact expected output?
3. Is there a description of how test cases will be selected to include best, worst, typical, and corner cases?
4. Is there a traceability matrix tying test cases to requirement and specification items?
5. Do you have the small details covered such as cover page with names, contact information, version, and date? Also, do you have page numbers on all pages?

Problem #4

Answer the following questions in one or two sentences each. All of these questions come from MMM.

- a) According to Brooks, what is the most important consideration in system design?

Conceptual integrity (page 42).

2 pts for each subproblem, except 3 pts for f and g.

- b) Why did the Tower of Babel fail?

Lack of communications and its consequent, organization (page 74)

- c) What is the “Documentary Hypothesis”

“The hypothesis: Amid a wash of paper, a small number of documents become the critical pivots around which every project’s management revolves. These are the manager’s chief personal tools” (page 74)

- d) Usually, how do schedules slip?

“Usually, however, the disaster is due to termites, not tornadoes.” (page 154). That is, schedules slip one day at a time.

- e) What does Brooks think about flowcharts?

“The flow chart is the most thoroughly oversold piece of program documentation.” (page 167).

- f) What does Brooks think about object oriented programming?

“Many students of the art hold out more hope for object-oriented programming than for any of the other technical fads of the day. I am among them.” (page 189)

- g) What does Brooks think about the complexity of software systems as compared to other things that humans make?

“Software systems are perhaps the most intricate and complex (in terms of number of distinct kinds of parts) of the things humanity makes.” (page 250)

Problem #5

MMM and the No Silver Bullet article were both written before offshoring became a reality. As you know, offshoring is the sending of work to other countries where labor costs are lower than in the US. Is offshoring a silver bullet? Why or why not? Under what precise conditions could we consider offshoring to be a silver bullet? Is this reality?

A Silver Bullet is something that can drop the production cost of software by magnitudes. So, if a foreign country could be found where software developers make 10x less than in the US, are equally competent to US software developers, and communications between developers was not an issue... then offshoring could indeed be called a “Silver Bullet”. To my understanding, the cost of a developer in India or China is not 10x less than here (about 4x less), they are generally less competent than US developers, and communications is a major issue. So, offshoring cannot be a Silver Bullet.

Understand what is a silver bullet = 10 pts. Analysis = 6 pts.

Problem #6

What are the three most valuable/useful things you learned in the guest talks? Identify who was the guest speaker for each item you list.

Multiple answers are possible.

5 pts for each thing. Points off if trivial or don’t know who guest speaker was.

Extra Credit

Write a function that takes a string containing numbers separated by commas as its only argument and returns the smallest number in the string as an integer. The length of the string is not previously known.

Example #1:

Input string: "12,4,2,34,4,6,6,193"

Return value: 2

Example #2:

Input string: "3,3,-2,1,123,-5,6,12"

Return value: -5

Explain how you would test this function.

Why this problem? This is the **exact problem** given to a previous Senior Project student when he interviewed at Microsoft last week. Microsoft interviews consist of "go to the board and write the code for..." kind of questions. Yes, he wrote the code, explained how to test the function, and indeed got the job. He had about 10 minutes to solve this problem. So, are you ready for an interview at Microsoft? :-).

5 pts for code (strtok() and atoi() are key) and 5 pts for testing where invalid inputs are key.

Here is a solution. Key to the solution is the use of strtok() and atoi().

```
//===== file = strTest.c =====
//= Test scaffold for str2MinInt() =
//=====
//= Notes: =
//= 1) Input from standard input (input must be quoted bounded) =
//=-----=
//= Example execution: =
//= strTest "3,5,2,9,0,-1,10" =
//= Smallest integer is: -1 =
//=-----=
//= Build: bcc32 str2minint.c =
//=====
#include <stdio.h> // Needed for printf()
#include <string.h> // Needed for strtok()
#include <stdlib.h> // Needed for atoi()

// Function prototype
int str2MinInt(char *cStr);

//=====
//= main program =
//=====
int main(int argc, char *argv[])
{
    printf("Smallest integer is: %d\n", str2MinInt(argv[1]));

    return 0;
}

//-----
//- Returns smallest integer value in a comma delimited string of integers -
//-----
//- Input: pointer to a string -
//- Return: integer value -
//- Side effects: none -
//-----
//- Assumptions: -
//- 1) It is assumed that all integer values are within MAXINT and -
//- MININT bounds -
//- 2) If the input string contains any characters other than digit, -
//- space, or comma the function will return zero -
```

```

//-----
// Known bugs:
// 1) Error checking for values outside of MAXINT and MININT bounds not
//    implemented. Values outside of these bounds may affect the
//    correctness of this function.
// 2) Not possible to distinguish between zero as valid smallest value
//    and an input string containing invalid characters
//-----
int str2MinInt(char *cStr)
{
    int iNum, iTemp;    // Local integer variables
    char *ptrStr;      // Local pointer variable

    //Get pointer to first substring in input
    ptrStr = strtok(cStr, ",");

    // Convert the substring to an integer value
    iNum = atoi(ptrStr);

    // Return if it is zero
    if (iNum == 0) return 0;

    // Check each remaining substring in the input
    while((ptrStr = strtok(NULL, ",")) != NULL)
    {
        // Convert the substring to an integer value
        iTemp = atoi(ptrStr);

        // Return if it is zero
        if (iTemp == 0) return 0;

        // Compare save the smallest in iNum
        if(iTemp < iNum)
            iNum = iTemp;
    }

    //Return smallest value
    return iNum;
}

```

To test the code we need both valid and invalid input strings. Good invalid cases include:

- 1) Input contains non digits (say, "lbc"). Function should ignore such "numbers".
- 2) Input contains spaces. Function should skip over spaces.
- 3) Input contains values smaller than MININT and/or larger than MAXINT. Function should ignore such "numbers"