

Computer Science II: Programming

Task Analysis

1 Students evaluate the tasks a computer program is to perform.

- 1 Differentiate between the different tasks a computer program should perform **CS2P-1.1**
 - 2 Formulate solutions to the different tasks a computer program should perform **CS2P-1.2**
 - 3 Interpret prior solutions; In case prior code and design could be reused **CS2P-1.3**
-

Problem Analysis

1 Students analyze a problem and develop an advanced solution by creating a computer program.

- 1 Recognize and explain how to use a computer program to solve a problem **CS2P-2.1**
 - 2 Construct interactive computer programs that accept various forms of input and produce various forms of output, as a solution to an advanced computer programming problem **CS2P-2.2**
 - 3 Use print charts, file layouts, program narratives, hierarchy charts, and system flowcharts, which accurately depict the problem assigned and describe the solution **CS2P-2.3**
 - 4 Justify what programming methodology to use—object oriented or procedural **CS2P-2.4**
 - 5 Appraise the program schematics and usage; document the program and describe its use **CS2P-2.5**
 - 6 Recognize and explain the standard program flowchart symbols and use them correctly within the context of the basic control structures of sequence, selection and looping **CS2P-2.6**
-

Software Tools

1 Students apply and adapt software tools to develop an advanced computer program.

- 1 Construct an advanced program that processes information **CS2P-3.1**
- 2 Identify programming languages as procedural and object oriented **CS2P-3.2**
- 3 Recognize and explain class in object oriented programming **CS2P-3.3**
- 4 Recognize and explain object in object oriented programming **CS2P-3.4**

- 5 Recognize and explain method in object oriented programming CS2P-3.5
 - 6 Recognize and explain instance variable in object oriented programming CS2P-3.6
 - 7 Recognize and explain polymorphism in object oriented programming CS2P-3.7
 - 8 Recognize and explain inheritance in object oriented programming CS2P-3.8
 - 9 Recognize and explain overwriting methods in OOP CS2P-3.9
 - 10 Recognize and explain encapsulation in computer programming CS2P-3.10
 - 11 Apply and adapt at an advanced level fundamental programming concepts, including data types, control structures, methods, and arrays CS2P-3.11
 - 12 Develop advanced programs using reusable modules (modularization) CS2P-3.12
 - 13 Use advanced debugging techniques to correct and validate the computer program CS2P-3.13
 - 14 Construct the program in a high-level programming language based on a created design CS2P-3.14
 - 15 Determine how to integrate a computer program with a web browser CS2P-3.15
 - 16 Determine how to use a common code/ GUI library CS2P-3.16
 - 17 Identify controls (push buttons, entry fields, etc.), their properties, methods, and when to use each control CS2P-3.17
-

Algorithms

1 Students design a solution to the problem using algorithms.

- 1 Develop advanced algorithms to solve a computer programming problem(s) CS2P-4.1
 - 2 Apply and adapt math operators in a computer program CS2P-4.2
 - 3 Prescribe the use of algorithms to provide a solution to a programming problem CS2P-4.3
 - 4 Use pseudo code to describe a solution to an advanced programming problem CS2P-4.4
 - 5 Create a program flowchart and ANSI standard flowcharting symbols to define a solution to an advanced programming problem CS2P-4.5
 - 6 Explain how the algorithm can be used to solve a problem CS2P-4.6
-

Program Development

1 Students create an advanced functional computer program.

- 1 Define the process of programming. For example: STAIR, Statement, Tools, Algorithm, Implement, and Refine CS2P-5.1
- 2 Create an advanced computer program that corresponds to an algorithm or proposed solution CS2P-5.2
- 3 Demonstrate programming structures CS2P-5.3
- 4 Appraise the use of data variables and constants CS2P-5.4
- 5 Appraise the use of local and global scope CS2P-5.5

- 6 Appraise the use of conditionals (IF statements) CS2P-5.6
 - 7 Appraise the use of loops (while statements, for statements) CS2P-5.7
 - 8 Use single and multidimensional Arrays CS2P-5.8
 - 9 Create programmer defined functions and methods to break down the program logic and support reuse CS2P-5.9
 - 10 Define the graphical user interface CS2P-5.10
 - 11 Identify the parts of the programming platform CS2P-5.11
 - 12 Identify different types of errors and handle them programmatically CS2P-5.12
 - 13 Use the order of operations when using calculations CS2P-5.13
 - 14 Construct an advanced computer program using proper condition and loop techniques CS2P-5.14
 - 15 Use correct naming conventions in variable declarations, function declarations, class declarations, and other CS2P-5.15
-

Program Verification and Debugging

1 Students prove that an advanced computer program solution works by using verification and debugging techniques.

- 1 Predict and explain output CS2P-6.1
 - 2 Identify cause/effect for input/output CS2P-6.2
 - 3 Perform input validation CS2P-6.3
 - 4 Scrutinize peers code for errors CS2P-6.4
 - 5 Show the use of proper internal documentation and coding comments CS2P-6.5
-

Documentation

1 Students connect the associated task with the code by providing documentation.

- 1 Describe the function of an advanced computer program CS2P-7.1
- 2 Identify the purposes of an advanced computer program CS2P-7.2
- 3 Explain concepts related to an advanced computer program CS2P-7.3
- 4 Evaluate how to use an advanced computer program CS2P-7.4
- 5 Identify cause/effect by explaining input and output related to an advanced computer program CS2P-7.5
- 6 Interpret input/output of an advanced computer program CS2P-7.6