COMPUTER PROGRAMMING 2
(NEW NAME: INTERMEDIATE COMPUTER PROGRAMMING 2022-23)
COURSE CODE: 5051

COURSE DESCRIPTION: Intermediate Computing Programming, formerly known as Computer Programming 2, is designed to expand upon the fundamental programming skills acquired in Introduction to Computer Programming (Computer Programming 1). Topics include intermediate program design and development techniques, security and ethics, and practical experience in programming using a modern, text-based programming language.

OBJECTIVE: Given the necessary equipment, software, supplies, and facilities, the student will be able to successfully complete the following core standards for courses that grant one unit of credit.

PREREQUISITE: Computer Programming 1

COMPUTERS REQUIRED: One computer per student

CREDIT: 1 unit (120 hours)

RECOMMENDED GRADE LEVEL: 9-12

A. SAFETY

Proficient professionals know the academic subject matter, including safety as required for proficiency within their area. They will use this knowledge as needed in their positions. The following accountability criteria are considered essential for students in any program of study.

1. Review school safety policies and procedures.
2. Review classroom safety rules and procedures.
3. Review safety procedures for using equipment in the classroom.
4. Identify major causes of work-related accidents in office environments.
5. Demonstrate safety skills in an office/work environment.

B. STUDENT ORGANIZATIONS

Proficient professionals know the academic subject matter, including professional development, required for proficiency within their area. They will use this knowledge as needed in their positions. The following accountability criteria are considered essential for students in any program of study.

1. Identify the purpose and goals of a Career and Technology Student Organization (CTSO).
2. Explain how CTSOs are integral parts of specific clusters, majors, and/or courses.
3. Explain the benefits and responsibilities of being a member of a CTSO.
4. List leadership opportunities that are available to students through participation in CTSO conferences, competitions, community service, philanthropy, and other activities.
5. Explain how participation in CTSOs can promote lifelong benefits in other

professional and civic organizations.

C. TECHNOLOGY KNOWLEDGE

Proficient professionals know the academic subject matter, including the ethical use of technology as needed in their positions. The following accountability criteria are considered essential for students in any program of study.

1. Demonstrate proficiency and skills associated with the use of technologies that are common to a specific occupation.
2. Identify proper netiquette when using e-mail, social media, and other technologies for communication purposes.
3. Identify potential abuse and unethical uses of laptops, tablets, computers, and/or networks.
4. Explain the consequences of social, illegal, and unethical uses of technology (e.g., piracy; illegal downloading; cyberbullying; licensing infringement; inappropriate uses of software, hardware, and mobile devices in the work environment).
5. Discuss legal issues and the terms of use related to copyright laws, Creative Commons, fair use laws, and ethics pertaining to downloading of images, photographs, Creative Commons, documents, video, sounds, music, trademarks, and other elements for personal use.
6. Describe ethical and legal practices of safeguarding the confidentiality of business-related information.
7. Describe possible threats to a laptop, tablet, computer, and/or network and methods of avoiding attacks.

D. PERSONAL QUALITIES AND INTERPERSONAL SKILLS

Proficient professionals know the academic subject matter, including positive work practices and interpersonal skills, as needed in their positions. The following accountability criteria are considered essential for students in any program of study.

1. Demonstrate creativity and innovation.
2. Demonstrate critical thinking and problem-solving skills.
3. Demonstrate initiative and self-direction.
4. Demonstrate integrity.
5. Demonstrate work ethic.
6. Demonstrate conflict resolution skills.
7. Demonstrate listening and speaking skills.
8. Demonstrate respect for diversity.
9. Demonstrate customer service orientation.
10. Demonstrate teamwork.

E. PROFESSIONAL KNOWLEDGE

Proficient professionals know the academic subject matter, including positive work practices and interpersonal skills, as needed in their positions. The following accountability criteria are considered essential for students in any program of study.

1. Demonstrate global or "big picture" thinking.
2. Demonstrate career and life management skills and goal-making.
3. Demonstrate continuous learning and adaptability skills to changing job requirements.
4. Demonstrate time and resource management skills.
5. Demonstrates information literacy skills.
6. Demonstrates information security skills.
7. Demonstrates information technology skills.
8. Demonstrates knowledge and use of job-specific tools and technologies.
9. Demonstrate job-specific mathematics skills.
10. Demonstrates professionalism in the workplace.
11. Demonstrates reading and writing skills.
12. Demonstrates workplace safety.

F. SECURITY & ETHICS

Programmers understand the significance of security and ethics. The following accountability-criteria are considered essential for students in the Computer Programming program of study.

1. Evaluate how sensitive data can be affected by malware and other attacks (e.g., denial-of-service attacks, ransomware, viruses, worms, spyware, phishing) (CSTA, 2017).
2. Identify best practices of software development that improve computer security and protect devices and information from unauthorized access (e.g., encryption, authentication strategies, secure coding, safeguarding keys) (CSTA, 2017)
3. Explain how to document programs that use non-user-created resources (e.g., code, media, libraries) giving attribution to the original creator. (Ethical and fair use)
4. Examine licenses (i.e., permissions) that limit or restrict use of resources (e.g., freeware, shareware, open source, creative commons)
5. Discuss the implications of using licensed resources in a developed solution.

G. INTERMEDIATE PROGRAM DESIGN & DEVELOPMENT

Programmers demonstrate effective programming design skills, including problem-solving, using data and control structures, debugging, and program documentation, to successfully design and run programs. The following accountability-criteria are considered essential for students in the Computer Programming program of study.

PROBLEM SOLVING & PROGRAM DESIGN

1. Investigate how a problem is systematically solved through the selection and integration of hardware and software components.
2. Justify hardware and software selections for specific applications by evaluating the components (e.g., databases, sensors, application programming interfaces (APIs)) of various computing devices (e.g., desktops, laptops, tablets, smartphones, and specialized devices like global positioning systems (GPSs)).
3. Compare and contrast ways software developers protect both devices and information from unauthorized access (e.g., encryption, authentication strategies, secure coding, safeguarding keys) (CSTA, 2017).

4. Evaluate existing computing solutions according to inclusivity or non-inclusivity (e.g., sight and hearing impairment, ethnicity, age).
5. Evaluate algorithms in terms of efficiency, correctness, and clarity (CSTA, 2017).
6. Categorize a variety of algorithms (e.g, linear, exponential, logarithmic, regression etc.).

DATA STRUCTURES

1. Compare and contrast the various data collection methods, data analysis tools, and data representation tools.
2. Compare and contrast the various data storage tools and data organization methods.
3. Compare and contrast the various data collection methods, data analysis tools, and data representation tools.
4. Determine when data structures (e.g., lists, arrays, tuples, stacks, queues, structures) are more appropriate than simple data types, and incorporate them into programs.
5. Create and modify data structures (e.g., lists, arrays, tuples, stacks, queues, structures) are more appropriate than simple data types, and incorporate them into programs.

CONTROL STRUCTURES

1. Justify the selection of specific programming constructs, explaining the benefits and drawbacks of choices made on the program's execution. (e.g., procedures, modules, objects).
2. Build a complex solution to a problem that incorporates reusable code (e.g., student created, application programming interfaces (APIs), libraries).
3. Justify the selection of specific control structures explaining the benefits and drawbacks of choices made (e.g., trade- offs involving implementation, readability, and program performance). (e.g., conditionals, loops, event handlers, recursion).
4. Implement and modify built-in classes.
5. Create and use user-defined classes.
6. Instantiate and use objects from user-defined classes.

TESTING, DEBUGGING & REVISIONS

1. Recommend modifications for existing computing devices and software to improve functionality for end users.
2. Evaluate and refine programs to make them more usable, functional, and accessible.

DOCUMENTATION

1. Implement version control to track program refinements. (Refers to using folder structures for version control. This is not specific to using configuration management software.)
2. Document programs that use non-user-created resources (e.g., code, media, libraries) giving attribution to the original creator.

Additional Course Materials and Resources

Academic Standards and Indicators

Computer Science Academic Standards and Indicators