COMPUTER PROGRAMMING 1 with SWIFT
COURSE CODE: 5066

COURSE DESCRIPTION: The Computer Programming 1 with Swift course is designed to emphasize the fundamentals of computer programming using Swift. Topics include computer software, program design and development, and practical experience in programming, using modern, object-oriented languages.

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:	Any computer-related course, Algebra 1 (or equivalent), and/ or Teacher recommendation
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. COMPUTING SYSTEMS

Programmers understand the significance of various computing systems. The following accountability-criteria are considered essential for students in the Computer Programming program of study.

1. Describe the hardware requirements needed to run operational systems and systems software. (ex. application software, database management, networking software, etc.).
2. Compare and contrast the elements of a computing system by examining hardware elements for their intended use (e.g., input-output (I/O) devices, random access memory (RAM), read only memory (ROM), storage devices, motherboards, and processors including the arithmetic logic unit (ALU), control unit, registers, cache memory, example implementations of some of these components using logic gates) (Virginia, 2017).
3. Describe the various data storage tools and data organization methods.
4. Research computing solutions to problems in different countries, considering the personal, ethical, social, economic, and cultural impact (e.g., the use of drones to deliver blood and medical supplies in countries in Africa, the use of Uber in India to address traffic congestion).

G. PROGRAM DEVELOPMENT AND DESIGN

Programmers demonstrate effective programming design skills, including algorithms, control and data structures, prototypes with review, and iteration to successfully design programs. The following accountability-criteria are considered essential for students in the Computer Programming program of study.

PROBLEM-SOLVING AND PROGRAM DESIGN

1. Document steps in the design process.
2. Develop an algorithm for a program using a design tool (e.g., pseudocode, flowcharts, human-language algorithm, etc.).
3. Construct data flows based on program requirements.
4. Utilize version control as a part of the design process to document revisions in an iterative development cycle.
5. Identify the advantages and disadvantages of diverse perspectives and backgrounds when solving computational problems.
6. Make use of standard programming control structures during algorithm design.
7. Analyze the sequence of instructions to determine if they will accomplish a task.

8. Create algorithms to solve computational problems that have an application in the real world (e.g., local community, church, civic organization, school, home life).
9. Adapt predefined algorithms to solve computational problems.
10. Select appropriate data types to store information used in the program.
11. Demonstrate an understanding of how to collect requirements.
12. Decompose tasks into smaller, reusable parts to facilitate the design, implementation, and review of programs.

DATA STRUCTURES

1. Create valid variables and constants using appropriate data types to store information used in the program.
2. Determine the scope and lifetime of variables (e.g., local, global, static).
3. Create valid variables and constants using appropriate scope (e.g., local, global, static).
4. Describe the properties of a data set that could be used to explore a real world phenomenon or support a claim.
5. Compare and contrast data sets that could be used to explore a real-world phenomenon or support a claim.
6. Create data sets that could be used to explore a real world phenomenon or support a claim.
7. Organize collected data to communicate the solution to a real-world phenomenon and support a claim.

CONTROL STRUCTURES

1. Summarize the differences of Sequential Programming and Event Driven Programming.
2. Develop an interactive program that includes features to get input and provide feedback/information (e.g., alerts, messages, input boxes).
3. Develop a program that correctly utilizes conditionals (if, else if, else, switch) to produce multiple outcomes based on input given from a user.
4. Develop a program that correctly utilizes the different Control structures (e.g., Sequence logic, Selection logic, iteration Logic) to basically analyze and choose in which direction a program flows based on certain parameters and conditions.
5. Develop a program that correctly utilizes Loops to produce multiple outcomes based on input given from a user.
6. Trace the flow of execution of a program that uses a combination of control structures (e.g., conditionals, loops, event handlers, recursion).
7. Design and iteratively develop programs that combine control structures (e.g., conditionals, loops, event handlers, recursion).
8. Trace the flow of execution of a program that uses a variety of programming constructs (e.g., procedures, modules, objects).
9. Design a solution through systematic analysis using programming constructs (e.g., procedures, modules, objects).
10. Explain different decision structures that control program flow.
11. Select from different looping/iteration structures that control program flow.

TESTING, DEBUGGING, AND REVISIONS

1. Plan and develop programs for a variety of audiences using a process that incorporates development, feedback, and revision.

   a. Review the program.
   b. Build the program.
   c. Execute the program to test the logical validity of an application program given appropriate data.
   d. Identify values of variables at different points in the flow of execution.
   e. Debug the program for errors (e.g., syntax and build errors).
2. Systematically test programs using a range of test cases to meet design specifications (e.g., specific outcomes, functionality, user interface, error handling).
   a. Develop a test strategy.
   b. Create a test plan.
   c. Design a test suite of conditions to evaluate best and worst cases of a program.
   d. Identify the difference between a test case and test script.
   e. Create a test script.
   f. Demonstrate the ability to debug the program for errors (e.g., run-time/exception, logic/semantic).
3. Develop code to solve the smaller parts of a decomposed task that can be reused to solve similar problems (e.g., procedures, functions, objects).
4. Seek and incorporate feedback to refine a solution (e.g., users, team members, code review, teachers).

Additional Course Materials and Resources

Academic Standards and Indicators

Computer Science Academic Standard and Indicators