



Project-based Final Evaluation: 40%

Course Identification

Name of program – Code	COMPUTER SCIENCE TECHNOLOGY – VIDEO GAME PROGRAMMING –420.BX
Course title:	STRUCTURED PROGRAMMING
Course number:	420-P16-AS
Group:	07152
Teacher's name:	Olivier Vézina
Duration:	Extended
Semester:	Fall 2020

Student Identification

Name: _____ Student number: _____

Date: December 8th 2020 Result: _____

☐ I declare that this is an original work, and that I credited all content sources of which I am not the author (online and printed, images, graphics, films, etc.), in the required quotation and citation style for this work.

Standard of the Evaluated Competency

Statement of the evaluated competency – Code

Use programming languages – 00Q2

Evaluated elements of the competency

1. Analyze the problem
2. Translate the algorithm into a programming language

Interact in a professional setting - 00SE

Evaluated elements of the competency

1. Establish professional relationships with users and clients

Instructions

- Permitted equipment: **everything**
- The teacher will not answer questions during the project.
- Plagiarism, attempts at plagiarism or complicity in plagiarism during a summative evaluation results in a mark of zero (0). In the case of recidivism, in the same course or in another course, the student will be given a grade of '0' for the course in question. (IPEL – Article 5.16).

Mark Breakdown

This evaluation is on 40 points, distributed as follows:

- | | |
|---|--------------------------|
| • Implementation plan | For a total of 10 points |
| • Presentation of the implementation plan | For a total of 05 points |
| • Implementation | For a total of 25 points |

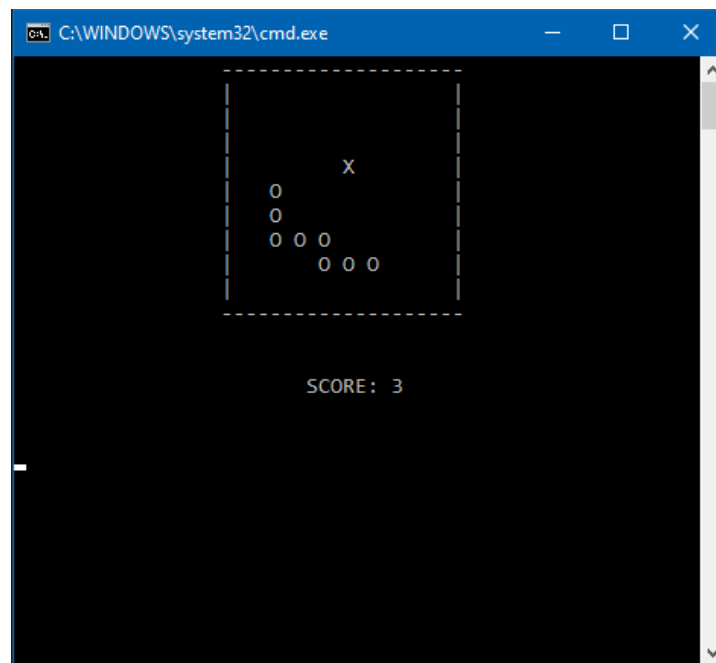
TOTAL: 40 POINTS

SNAKE C++ PROJECT

In this final evaluation, you are asked to implement a console program of the famous Snake game.

Here are the rules for our snake game:

- The game must be played on a 9 x 9 grid
- The snake starts at the middle of the grid (row index : 4 / column index : 4)
- The snake has a starting size of 5.
- An apple is spawned at a random location on the grid (but not on the snake)
- Pressing A moves the snake left by one cell
- Pressing D moves the snake right by one cell
- Pressing S moves the snake down by one cell
- Pressing W moves the snake up by one cell
- If the head of the snake touches the body, the game is over
- If the head of the snake exits the grid, the game is over
- If the head of the snake touches an apple, the snake's size increments by one and another apple is randomly spawned on the grid (but not on the snake)
- Every time the snake eats an apple, the score increments by one



Snake is a pretty **popular** game and a good stepping stone in your journey to become a programmer. Copying the **code of someone else** will result in the automatic failure of this class whether it be from the internet or from a peer.

This evaluation is made up of 3 distinctive parts.

PART 1: Implementation plan 10 points

The first part of this final evaluation consists of elaborating an implementation plan (analysis). Before coding, it is recommended to have a clear understanding of what the problem requires you to do. Hopefully, taking the time to write the implementation plan will greatly help you when you'll have to implement the solution.

A good implementation plan is made up of two sections, namely, the solution sequence and the implementation description.

1. Solution sequence section

The best way to solve a computer problem is to split it into smaller problems. Each small problem is a task that needs to be completed.

In this section, write the list of tasks that need to be completed to properly code the snake game. Each task should be briefly described. The more info you give, the better!

2. Implementation description

The implementation description is a bit more technical. In this section, you need to name and describe each function that you intend to use. In other words, your implementation description should contain the following information:

- The name of each function that you intend to use
- A brief description of the function's intended purpose
- The pseudocode for that function

The next two pages will help you get a better understanding of what's asked from you. Use the template available on LÉA.

Example of the solution sequence section for

ROCK PAPER SCISSORS in C++

Step 1: Welcome the player

To welcome the player, I'm going to print a welcoming message to the console

Step 2: Allow the player to select the playoff format of the game

The player has to select the playoff format. To do so, I will ask the user to enter a number from 1 to 3 values where:

- 1 -> Best of 3
- 2 -> Best of 5
- 3 -> Best of 7

Each option means that the game will require a different number of won rounds

Step 3: As long as the game isn't over

The concept of the game being over is tied to the number of won rounds. To determine if the game is over, I will have to check if the number of won rounds of both the human or the computer is equal to the previously selected playoff format requirement. Using a while loop will come in handy.

Step 4: Allow the player to select his or her hand

The player has to select his or her hand. To do so, I will ask the user to enter a number from 1 to 3 values where:

- 1 -> ROCK
- 2 -> PAPER
- 3 -> SCISSORS

I will store that decision so that it can be compared with the computer's decision later.

Step 5: Computer decision

The computer has to select its hand. To do so, I will generate a random number between 1 and 3 where:

- 1 -> ROCK
- 2 -> PAPER
- 3 -> SCISSORS

I will store that decision so that it can be compared with the player's decision later.

Step 6: ...

Example of the implementation description section for

ROCK PAPER SCISSORS in C++

//This function's purpose is to allow the player to select the playoff format of the game

void SelectFormat()

1. Print the playoff format options to the player
2. Read the player's playoff format option
3. If the player's option is 1, set the rounds to win to 2.
If the player's option is 2, set the rounds to win to 3
If the player's option is 3, set the rounds to win to 4

//This function's purpose is to check whether the end condition of the game has been reached. If a player has reached the requirement number of wins, the function returns true. Otherwise, the function returns false.

bool GameHasFinished()

1. If the human has won the required number of rounds, return true
2. If the computer has won the required number of rounds, return true
3. Otherwise, return false

//This function's purpose is to allow the player to select a hand in a round of Rock Paper Scissors

void HumanSelectHand()

1. Print the hand options to the player
2. Read the player's hand option
3. If the player's option is 1, set the player's current hand to ROCK
If the player's option is 2, set the player's current hand to PAPER
If the player's option is 3, set the player's current hand to SCISSORS

//This function's purpose is to randomly generate a number from 1 to 3 to represent the computer's decision

void ComputerSelectHand()

1. Generate a random number between 1 and 3
2. If the computer's decision is 1, set the player's current hand to ROCK
If the computer's decision is 2, set the player's current hand to PAPER
If the computer's decision is 3, set the player's current hand to SCISSORS

void ...

PART 2: One-on-one conversation 05 points

Even if programmers mainly work individually in front of a computer, they often discuss with their peers about the implementation process of a new feature or a new software. Furthermore, programmers that work for clients often have to meet with them and explain how they will fulfill the needs of said client.

In this section, you will have to present your implementation plan (part 1) in a one on one meeting of about 7 minutes with me. Expect a conversation that goes back and forth instead of a presentation.

PART 3: Implementation 25 points

In this section, you are asked to implement a game of Snake in C++ that can be played from the console.

Please refer to page 3 of this document for the rules.

Here's a demonstration of how the game should look like: <https://imgur.com/a/c4Wp5bt>

The best way to implement this game is to start with a snake that has only one part. Once you're satisfied with that, add the feature where the snake is made up of multiple parts.

Specific implementation requirements

- To represent a coordinate on the grid, use a struct called **Coordinate** that holds two properties. These properties should represent the **row** index and the **column** index
- To represent the snake, use a **vector** of **Coordinate** called **snakeParts**. Each snake part is represented as a **Coordinate**. When eating an apple, `push_back( )` a new coordinate in the vector. The coordinate of that new snake part should be the same as the head's coordinate.
- To represent directions, use an enum called **Direction**. Its content should be
LEFT,
RIGHT,
DOWN,
UP
- Use the file `snake.cpp`. This file is available on LÉA.

PART 1: Implementation plan

Element of competency: Analyze the problem (00Q2.1)					
Performance criteria	Highly satisfactory	Satisfactory	Unsatisfactory	Highly unsatisfactory	Total
1.1 Correct breakdown of the problem	5 All the required elements are present	4-3 Some elements are missing	2-1 Most of the elements are missing	0 Almost all the elements are missing	/10
1.2 Proper identification of input and output data and of the nature of the processes	5 Every step is properly described	4-3 Some steps are not properly described or missing	2-1 Most of the steps are not properly described or missing	0 Almost all the steps are not properly described or missing	

PART 2: One-on-one discussion

Element of competency: Establish professional relationships with users and clients (00SE.1)					
Performance criteria	Highly satisfactory	Satisfactory	Unsatisfactory	Highly unsatisfactory	Total
1.3 Appropriate choice of adaptation of the algorithm	5 Great communication of the problem and its requirements	4-3 Good communication of the problem and its requirements	2-1 The problem and its requirements aren't communicated well	0 The problem and its requirements are not the topic of the presentation	/05

PART 3: Implementation

Element of competency: Translate the algorithm into a programming language (00Q2.2)					
Performance criteria	Highly satisfactory	Satisfactory	Unsatisfactory	Highly unsatisfactory	Total
2.1 Appropriate choice of instructions and types of elementary data	5 Use of appropriate types and instructions throughout the entirety of the program	4-3 Some types and instructions in the program are not appropriate	2-1 Most of the types and instructions are not appropriate	0 Almost all of the types and instructions in the program are not appropriate	/25
2.2 Efficient modularization of code	5 Every major step in modularized efficiently	4-3 Some steps are not modularized efficiently	2-1 Most of the steps are not modularized efficiently	0 Almost all of the steps are not modularized efficiently	
2.3 Logical organization of instructions	5 Every line of code fits in the expected sequence of operations	4-3 Some lines of code are missing from the expected sequence of operations	2-1 Most of the lines of code are missing from the expected sequence of operations	0 Almost all the lines of code are missing from the expected sequence of operations	
2.4 Compliance with the language syntax	5 Use of a valid C++ syntax throughout the entirety of the program	4-3 Some elements of the program don't respect the C++ syntax	2-1 Most of the elements of the program don't respect the C++ syntax	0 Almost all of the elements in the program don't respect the C++ syntax	
2.5 Computer code consistent with the algorithm	5 The snake game is perfectly implemented	4-3 Most of the snake game is implemented	2-1 Most of the snake game's implementation is missing	0 Almost all of the snake game's implementation is missing	