VCE Computing

# Unit 2 - SAC 4

## Outline

Design and develop a solution to a problem using a programming language.

## Submission

The SAC is made up of two parts:

1. Design - Pseudocode, Data Dictionary, Object Description, Mockup
2. Development - Manipulation, Validation, Testing, Documentation

* The design documents must be created using Google Docs, and saved using this naming convention - YourName - U2O1 SAC 4 - Name of document.
* The design and testing documents (exported as a .docx) and the C# solution (as a zip) must be uploaded to the appropriate Learning Task on Compass by the due date.

## Marking

* If you have not submitted all coursework / homework by the **end of the SAC**, the **maximum** mark you can receive is an **E**.
* This SAC will be marked out of 20 and uses [this rubric](https://drive.google.com/open?id=1fjIiWSQtZgp-xZq-uSa14YBkOtdLMWbUhfdm7ZDw5UY).
* The following scale is used to mark the SAC:

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| % | 0 | 10 | 15 | 30 | 40 | 50 | 60 | 65 | 70 | 80 | 90 |
| Mark | UG | E | E+ | D | D+ | C | C+ | B | B+ | A | A+ |

## Time allocation

You have 9 periods to complete the SAC. It MUST be completed during class time only.

## Resources

You are free to use any resources to complete this SAC, apart from work completed by other students.

## Late Submission

If you submit the SAC after the due date, the maximum mark you can receive is an E. Extensions MUST be organised prior to the due date.

## Redemption Policy

Students that are found to have plagiarised, copied, or otherwise submitted work that is not their own, will receive a Fail grade for the SAC and be awarded an N for Unit 2. Students wishing to pass this unit will then be required to complete a redemption SAC.

## Task: Rock, Paper, Scissors

Create a computerised version of the game [*Rock, Paper, Scissors.*](https://en.wikipedia.org/wiki/Rock%E2%80%93paper%E2%80%93scissors)

**Description:**

The user and the computer chooses either rock, paper or scissors. The items are compared, and whichever player chooses the more powerful item wins. If the same items are chosen, the game ends in a tie.

Before development can begin, the **design tools** must be completed. This includes:

### MockUp

Draw a mockup (paper or Google Drawing) of what the form will look like. The user can use either radio buttons or a textbox to enter their choice. Labels must be included to display headings, the user and computer choice, and the game outcome. Buttons must be used to play the game, and to reset all controls to their default state.

### Object Description

Complete an object description (Google Doc) for each control placed on the form.

### Data Dictionary

Complete a data dictionary detailing all variables used by the program:

* userChoice
* comptuerChoice

### Pseudocode

Complete the Pseudocode for all methods based on the descriptions provided below.

**playButton\_Click()** - This method is created by double clicking on the button

Call **playGame()**

**resetButton\_Click()** - This method is created by double clicking on the button

Call **resetGame()**

**playGame()** - Self made method

Create a variable called **userChoice**, set equal to **getUserChoice()**

Create a variable called **computerChoice**, set equal to **getComputerChoice()**

Call **determineWinner()** and pass it **userChoice** and **comptuerChoice**

**resetGame()** - Self made method

Resets all **labels** and **variables** back to their original state.

**getUserChoice()** - Self made function

Gets and **returns** the user's choice from a textbox or radio button.

**getComputerChoice()** - Self made function

Generate a random integer between and including 0 and 2.

Depending on the number **return** rock, paper or scissors.

**determineWinner()** - Self made function

Two parameters to be passed - **userChoice** and **comptuerChoice**

Compare both choices for all outcomes and **return** the result.

### Development (all activities within the PSM)

#### Manipulation

Complete the solution in C#.

#### Validation

Ensure the application does not crash and only allows the user to enter valid items.

Testing

Complete a Testing table.

#### Documentation

Ensure all methods are commented to describe what each one does.

## Appendix

C# to generate a random integer between 0 and 2.

Random rnd = new Random();

int randomNumber = rnd.Next(0, 3); // Random no. >=0 and < 3

Make sure to include an **else** statement that **returns null** at the end of **all if statements**.