Module 1 Computer Programming

Module title	Computer Programming
Module NFQ level (only if an NFQ level can be demonstrated)	n/a
Module number/reference	BSCH-CP
Parent programme(s)	Bachelor of Science (Honours) in Computing Science
Stage of parent programme	Stage 1
Semester (semester1/semester2 if applicable)	Semester 1 & 2
Module credit units (FET/HET/ECTS)	ECTS
Module credit number of units	10
List the teaching and learning modes	Direct, Blended
Entry requirements (statement of knowledge, skill and competence)	Learners must have achieved programme entry requirements.
Pre-requisite module titles	None
Co-requisite module titles	None
Is this a capstone module? (Yes or No)	No
Specification of the qualifications (academic, pedagogical and professional/occupational) and experience required of staff (staff includes workplace personnel who are responsible for learners such as apprentices, trainees and learners in clinical placements)	Qualified to as least a Bachelor of Science (Honours) level in Computer Science or equivalent and with a Certificate in Training and Education (30 ECTS at level 9 on the NFQ) or equivalent.with a Certificate in Training and Education (30 ECTS at level 9 on the NFQ) or equivalent.
Maximum number of learners per centre (or instance of the module)	60
Duration of the module	Two Academic Semesters, 24 weeks teaching
Average (over the duration of the module) of the contact hours per week	5
Module-specific physical resources and support required per centre (or instance of the module)	One class room with capacity for 60 learners along with one computer lab with capacity for 25 learners for each group of 25 learners

Analysis of required learning effort		
	Minimum ratio teacher / learner	**Hours**
Effort while in contact with staff		
Classroom and demonstrations	1:60	36
Monitoring and small-group teaching	1:25	72
Other (specify)		
Independent Learning		
Directed e-learning		
Independent Learning		82
Other hours (worksheets and assignments)		55
Work-based learning – learning effort		
Total Effort		250

Allocation of marks (within the module)					
	Continuous assessment	**Supervised project**	**Proctored practical examination**	**Proctored written examination**	**Total**
Percentage contribution	70%			30%	100%

Module aims and objectives

The aim of the module is to teach the learner how to design high-quality computer programs in a systematic way. All the relevant concepts and techniques are explained and exemplified in the clearest, simplest language. The objectives are to facilitate the learner to understand the theory underlying programming as a concept and to enhance the logical step by step approach to problem solving required. The basic concepts of command sequences, iteration and selection are introduced, and the constructs used in a modern programming language to implement these.

Minimum intended module learning outcomes

On successful completion of this module, the learner will be able to:

1. Define precisely the syntax and semantics of a programming language
2. Implement basic algorithms and data structures using a programming language
3. Verify the implementation of a given program by testing various inputs against expected outputs
4. Prepare the text of a program in a well-formatted, conventional manner and develop these programs using an integrated development environment
5. Select an appropriate program construct (or datatype) to achieve a given task
6. Document accurately the design of a program on-the-fly
7. Determine the basic efficiency of an algorithm

Rationale for inclusion of the module in the programme and its contribution to the overall MIPLOs

Computer programming is a fundamental skill in computing science. No matter what area of ICT a practitioner works in an understanding of programming concepts and implementation issues is essential. Appendix 1 of the programme document maps MIPLOs to the modules through which they are delivered.

Information provided to learners about the module

Learners receive a programme handbook to include module descriptor, module learning outcomes (MIMLO), class plan, assignment briefs, assessment strategy and reading materials.

Module content, organisation and structure

Introduction to problem solving

- How do you complete a task?
- Identifying sub-components on larger task
- Defining order of subcomponents
- Creation of algorithms
- Stepwise design of programs
- Verifying outcomes through testing

Introduction to programming

- Expressions and statements
- Basic arithmetic
- Comments
- Variables and assignment (Integers, doubles, Booleans, characters)
- Boolean expressions and logic
- Conditional statements
- Iteration statements
- User input
- Output
- String manipulation
- Sub-routines (Parameters, Signature, Procedures, Copy rule.)
- Arrays
- 2D arrays
- Objects as custom datatypes
- Access modifiers
- Object arrays
- Text File I/O
- Binary File I/O
- Records
- Simple sorting algorithms (selection sort, bubble sort, insertion sort)
- Binary search

Professional Practice

- Developing a good coding style
- Using comments effectively
- Naming conventions
- Indentation
- Code structure
- Testing and testing strategies

Module teaching and learning (including formative assessment) strategy

The module is taught as a combination of lectures and lab sessions. The lecture sessions discuss and explain to learners the theoretical underpinnings of how a Computer Programming works and functions along with best practices for designing programs. The practical lab sessions give learners an opportunity to interact with a development environment and learn the foundational skills of computer programming.

Assessment is divided into seven elements. First there are a number take home assignments that assess the learner's competency in specific areas of the syllabus, while giving them increasingly larger problems to solve. There will be two class tests in the second semester to assess the learner's understanding of the concepts covered throughout the module. Finally, there is an end of semester exam that tests the learners understanding of the theoretical material.

Timetabling, learner effort and credit

The module is timetabled as two 1-hour lectures and two 1.5-hour labs per week.
Continuous assessment spreads the learner effort to focus on small steps before integrating all steps into the overall process of computer program design and implementation.

There are 120 contact hours made up of 48 lectures delivered over 24 weeks with classes taking place in a classroom. There are also 48 lab sessions delivered over 24 weeks taking place in a fully equipped computer lab. The learner will need 75 hours of independent effort to further develop the skills and knowledge gained through the contact hours. An additional 55 hours are set aside for learners to work on worksheets and assignments that must be completed for the module.
The team believes that 250 hours of learner effort are required by learners to achieve the MIMLOs and justify the award of 10 ECTS credits at this stage of the programme.

Work-based learning and practice-placement

There is no work based learning or practice placement involved in the module.

E-learning
The college VLE is used to disseminate notes, advice, and online resources to support the learners. The learners are also given access to Lynda.com as a resource for reference.

Module physical resource requirements
Requirements are for a classroom for 60 learners equipped with a projector, and a 25-seater computer lab for practical sessions with access a development environment (Notepad++) and the Java SDK (this may change should better technologies arise).

Reading lists and other information resources
Recommended Text
Deitel, P. J. and Deitel, H. M. (2018) *Java: How to Program*. New York: Pearson.

Schildt, H. (2014) *Java: the Complete Reference*. San Francisco: McGraw-Hill.

Secondary Reading:
Charatan, Q. and Kans, A. (2009) *Java in Two Semesters*. London: McGraw-Hill.

Dos Reis, A. J. and Dos Reis, L. L. (2012) *An Introduction to Programming Using Java*. Sudbury: Jones & Bartlett Learning.

Specifications for module staffing requirements
For each instance of the module, one lecturer qualified to at least Bachelor of Science (Honours) in Computer Science or equivalent, and with a Certificate in Training and Education (30 ECTS at level 9 on the NFQ) or equivalent.with a Certificate in Training and Education (30 ECTS at level 9 on the NFQ) or equivalent.. Industry experience would be a benefit but is not a requirement.

Learners also benefit from the support of the programme director, programme administrator, learner representative and the Student Union and Counselling Service.

Module Assessment Strategy

The assignments constitute the overall grade achieved, and are based on each individual learner's work. The continuous assessments provide for ongoing feedback to the learner and relates to the module curriculum.

No.	Description	MIMLOs	Weighting
1	A series of take home assignments that will foster development of the skills of: • Program design • User input • Variable use • Conditional statements • Loops • String manipulation • File access • Objects.	1-7	50%
2	2 class tests with will assess a subset of the content for this module in a similar environment to the final exam. Class test 1 will assess: Variables, user input, if statements, loops, and String manipulation Class test 2 will assess: Arrays, objects, and file access	1, 2, 3, 6, 7	20%
3	Written exam that tests the theoretical aspects of the module	1, 2, 3, 6, 7	30%

All repeat work is capped at 40%.

Sample assessment materials

Note: All assignment briefs are subject to change in order to maintain current content.

Workbook 1: Variables, Conditional Statements and Iteration

Remember to include a comment at the start of each file with you name and student number

Question 1 (week 5)

Make a program to has three integers (d, h, and m), and display the number of seconds in d days, h hours, and m minutes (assume d, h, and m have values of 1 , 17 , 34 respectively).

Question 2 (week 5)

Make a program to swap the values of variables x and y.

Question 3 (week 6)

Create a program that takes an integer from the user and outputs just the last digit in the input value

Question 4 (week 6)

Create a program that takes 5 integer values from the user and prints them to the screen in reverse order to how they were entered.

Question 5 (week 7)

Make a program to input an integer representing a month number and output the name of the month. If the integer is not a month number, a suitable reproach message should be displayed.

Question 6 (week 7)

Design a program to input three positive integers representing day, month and year, and output 'yes' or 'no' according to whether they form a legitimate date.

For Example:

```
Input ➔      day = 10
             month = 5
             year = 1912
Output ➔     yes
Input ➔      day = 30
             month = 2
             year = 1958
Output ➔     no
```

Question (week 8)

Create a guessing game program. Your program should generate a random number between 1 and 100 (inclusive) and then ask the user for a guess. The game will keep asking for a guess until one of the following conditions are matched:

- The user guesses the number correctly (Win state)
- The user excesses 20 incorrect guesses (Lose state)
- The user enters a negative number (Quit state)

In addition, your program should do the following:

1. at the end, the computer prints the number of guesses made by the user,
2. each time the user guesses incorrectly, the computer says “too high” or “too low”

Workbook 2: String manipulation and Methods

<u>Remember to include a comment at the start of each file with you name and student number</u>

Question 1

Create a program to compare 3 letter variables and output the largest.

Question 2

Create a program to increase the letter in your name by 1. E.g. Eoin = Fpjo

Question 3

Create a program take a String from the user. If the length is greater than 10 then print “Double” to the screen, otherwise print “Single”.

Question 4

Design a function **distance** that takes four fractional number parameters x0, y0, x1, and y1, and returns the distance between points (x0, y0) and (x1, y1). (Look up the formula if you can’t remember it!)

Question 5

Design a procedure **printString** to input an integer n and a string s, and print n copies of (the value of) n on the screen, two copies per line (except, possibly, the last line).

If the user enters a negative value for n, the computer should print a complaint.

Question 6

Design a function **numTxt** that takes an integer n as parameter and returns a string.

The value of n is assumed to be in the range 0…9, and the value returned is its equivalent textual form. For example, the value returned by **numTxt**(4) should be "four".

Question 7

Design a method (called **encode**) to accept a string ***s***, composed exclusively of lowercase
alphabetic characters and spaces from the keyboard. The method should also accept a
positive integer ***n***. Your code should then apply a Caesar Cipher to the string and print the
encoded string.

You should also create a test program to verify the method you have created.

Tips: A Caesar Cipher is applied individually to each letter in the string. Each letter must be shifted forward **n** steps in the alphabet. If a letter is shifted off the end of the alphabet (i.e. past 'z'), then it is shifted all the way back around to the beginning of the alphabet (to 'a'). Spaces are left unchanged during the encoding process.

Examples: encode("hello world",1) will return "idmmp xpsme"
encode("hello world",2) will return "jennq yqtnf"
encode("hello world",3) will return "kfoor zruog"
encode("hello world",4) will return "lgpps asvph"

Assignment 1 Title: **Griffith College Fruit Machine**

For this assignment you are to create a complete game to emulate a fruit machine (http://en.wikipedia.org/wiki/Slot_machine). Your machine will use the number 1-7 to represent the outputs on the wheels.

The game should allow a user to begin with 10 credits, each time they play the game it uses 1 credit.

If they win a prize (combinations detailed below) the winning amount of credits should be added to their total (less the 1 credit they spent to play the game.

If they lose the game their total credits remaining should decrease by 1.

After each play the user should be presented with the following menu:

1. Display remaining credits
2. Play again
3. Exit game

Depending on what input the user enters the game should respond appropriately.

If the user selects to display credits, the program should show remaining credits and print the above menu again.

If the user selects to play game, the program should play one spin of the fruit machine and update the credits based on result.

If the user selects to exit game, the program should show remaining credits and then end the program.

If the user uses all of their credits a game over message should be displayed

Winning Payouts:

7-7-7	150 Credits
3 matching numbers (not 7-7-7)	80 Credits
3 in a row sequential order (i.e. 1-2-3)	40 Credits
3 in a row non sequential order (i.e. 2-1-3)	25 Credits
2 Matching even	10 Credits
2 Matching odd	5 Credits
Any one 7 (i.e. X-7-X)	2 Credits

Assessment Criteria

Your code will be assessed as follows:

Menu System: (10%)

- Display credits (5%)
- Play game (3%)
- Exit program (2%)

Game System: (55%)

- “Spin the wheels” (5%)
- Print result of spin (3%)
- Jackpot Check (5%)
- Match 3 (8%)
- 3-in-a-row (incrementing sequential) (7%)
- 3-in-a-row (non-sequential) (10%)
- 2 matching even (5%)
- 2 matching odd (5%)
- 7 plus 2 (3%)
- Using play credit (2%)
- Adding win credits (2%)

User Interface: (10%)

- Game style layout (6%)
- User feedback (4%)

Coding style: (20%)

- Indentation (3%)
- Comments (4%)
- Variable naming and use (3%)
- Code efficiency (10%)

Extra Features (5%)

- Addition of any extra features to enhance the game.

Common Sense Grading:

- Do not try and “game” the grading. If your code does not satisfy the basic requirements of the assignment you will <u>NOT</u> be awarded a passing grade.
- Code that does not compile will receive a penalty of 20% removed from grade (i.e. Grade 55% - penalty 20% = grade 35%.)

Assignment 2 Title: **Java-based Word Reversing Program**

Create a Java program to take a string S and an integer N from the user. The string is expected to be a series of words. Your program should pass through the String and reverse every other word.

Example

Input: “Hello World, My name is Eoin Carroll”, 2
Output: “Hello dlroW, My eman is nioE Carroll”

NOTE: This is to be accomplished without using String.reverse() or String Tokens.

Expected Functionality.

- The Program should reverse every nth word from the input string.
- The Program should contain a **reverse** method to do the reversing work.
- The Program output a suitable response if the nth word does not exist.
  - For Example : "Hello World", 3 ←there is no 3rd word.
- The Program should identify if the reversed word is an palindrome.
  - https://en.wikipedia.org/wiki/Palindrome

Extra Functionality

- The Program should identify if the user entered a pangram.
  - https://en.wikipedia.org/wiki/Pangram

Deliverables:

For this assignment you are required to submit 2 pieces of work:

1. Java file containing your program (70%)
2. A flow chart to show your design of the program (30%)

Grading Outline:

- Input = 5%
- Output = 5%
- String processing = 25%
- Word reversing = 20%
- Use of methods = 15%
- Check to see if reverse is possible = 5%
- Check for palindrome = 5%
- Check for pangram = 5%
- Structure and neatness of code = 15%

Assignment 3 Title: **Battleship Java Program**

Brief

For this assignment you are to build a version of Battleship (http://en.wikipedia.org/wiki/Battleship_(game)).

This assignment is a team assignment. You are required to create teams of 2. Each element of functionality is required to be assigned to a team member and documented as such in the comments of the code. Both team members MUST contribute to the codebase.

Rules are available here (https://www.thespruce.com/how-to-play-battleship-411069)[1] and a flash version of the game is available here (http://www.miniclip.com/games/battleships/en/) [2]

The program will use 2 2D Arrays (one for each player) and 2 arrays of boat objects (1 for each player).
The boat object needs to hold the following information:

- Boat type
- Boat size
- Hits made

Type of ship	Size
Aircraft Carrier	5
Battleship	4
Destroyer	3
Submarine	3
Patrol Boat	2

The Basic Game Requirements (75% of assessment):

1. Battleship is played on a 10x10 grid.
2. Your program should place both your boats and your opponent's boats on the grid automatically.
3. The boats cannot overlap.
4. Boats cannot be placed diagonally.
5. The program should print out (to the screen) your array (the position of your boats and where you opponents have launched missiles) and a grid of where you have launched missiles on your opponent's grid.
6. For the print out use the following legend:
   - 'B' for a boat.
   - 'H' for a hit.
   - 'X' for a miss.
   - ' '(a space character) for water.
7. Each player enters in a co-ordinate (for example (2,4)) and the program checks the opponent's grid and prints out a suitable response.
8. The program then reprints the 2 grids again (step 5).
9. The responses are as follows:
   - "Hit" when you hit an opponent's boat.
   - "Miss" when you hit water.
   - "Sunk" when the amount of hits on a boat is the size of the boat.
   - "You sunk my fleet" when all 5 boats are sunk.

[1] Last accessed 24/07/2018
[2] Last accessed 24/07/2018

Advanced Requirements (15% of assessment)

1. (10%) Allow for the player to manually input where each boat is:
    - The program should take 2 inputs; a start co-ordinate and a direction (down or across).
    - The program should check if the boat is big enough to fit on the grid without:
        - Going out over the edge of the board.
        - Overlapping with another boat.

2. (5%) Allow for the player to save the state of the game to a file and to load a saved game from a file.

Team Requirements (10% of assessment)

1. Accurate and clear documentation to show the responsibilities of each team member within the code.

Marking Criteria:

Well-formed, indented and commented code Implementation of all features listed. Neat, readable design.

Computer Programming Class Test 1

- This test consists of 3 questions, you are to complete any 2.
- You must show your design process for each question.
- The test will last 1 hour.

Question 1

Create a function to take a number N from the user and to output the Nth number in the Fibonacci sequence. For Example:

Fn = F(n-1) + F(n-2)

Input: 11

Output: 89

Note: The sequence begins 1,1

Question 2

Create a program to take a String str in from the user and process it as follows:

- Output the number of words
- Output the number of characters
- Output the number of sentences in the inputted String

Question 3

A string, consisting of a sequence of characters, both vowels and consonants:
    "my name is Jane Smith "

Design a program to input a string str and output the value of the most common character in the String. Here is *one example* of the I/O (user input underlined):

    Enter string: <u>my name is Jane Smith</u>
    Output: m

Note : you should ignore the case of the character (n==N)

Computer Programming Class Test 2

- This test consists of 3 questions, you are to complete any 2.
- You <u>must</u> show your design process for each question.
- The test will last 1 hour.

Question 1

Design a method to accept an integer array of 5 elements. Your program should find the minimum and maximum values that can be calculated by summing exactly four of the five integers. Then print the respective minimum and maximum values as a single line of two space-separated long integers.
Example:
If our array contains the following elements

2	4	6	8	1

We can calculate the following sums using four of the five integers:

- If we sum everything except 2, our sum is 4+6+8+1=19.
- If we sum everything except 4, our sum is 2+6+8+1=17.
- If we sum everything except 6, our sum is 2+4+8+1=15.
- If we sum everything except 8, our sum is 2+4+6+1=13.
- If we sum everything except 1, our sum is 2+4+6+8=20.

As you can see, the minimal sum is 2+4+6+1=13 and the maximal sum is 2+4+6+8=20.

In your answer clearly detail the design of the program as well as creating the code for the solution.

Question 2

Write a java class for a DVD. Each DVD has a title, a director and duration. You should also write a test program that creates and outputs the details of at least two DVDs by means of a **display** method.

In your answer clearly detail the design of the program as well as creating the code for the solution.

Question 3

Design a Java program to read a text file called **dwarves.txt**. The file contains the names of the 7 dwarves (Sleepy, Grumpy, Happy, Bashful, Doc, Dopey, Sneezy).
Your program should create a new text file for every dwarf in the source file. Each new file should be called after a dwarf.

The lines have no extra spaces or tabs at the beginning or end.
In your answer clearly detail the design of the program as well as creating the code for the solution.

GRIFFITH COLLEGE DUBLIN

QUALITY AND QUALIFICATIONS IRELAND EXAMINATION

COMPUTER PROGRAMMING

Lecturer(s):

External Examiner(s):

Date: XXXXXXXX **Time:** XXXXXXX

THIS PAPER CONSISTS OF FIVE QUESTIONS
FOUR QUESTIONS TO BE ATTEMPTED
SECTION A – COMPULSORY
SECTION B – THREE QUESTIONS TO BE ATTEMPTED

SECTION A – COMPULSORY

QUESTION 1

(a) Write Boolean expressions for the following (where A, B, C, D, E, F and G are integers):

(i) A is a negative odd number that is multiple of 53.

(ii) B is a number divisible by -17 and 39 but not 112.

(iii) C is either greater than 411 and less than 851 or C is -2342.

(iv) D is less than 20, E is not less than 0 and F is equal to 34.

(v) G is not a leap year.

(10 marks)

(b) On this course, we looked at While loops and For loops. Outline which structure would you use to make a loop that exits once a negative number is entered. In your answer explain why you made this choice.

(5 marks)

(c) Please explain the difference between the following 2 statements.

```
int x = 5;
x = 10;
```

```
int x = 5;
x == 10;
```

In your answer highlight any issues you can see with either of the two statements.

(5 marks)

(d) Explain the following error. In your answer highlight what causes it and how it can be avoided.

```
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 4
```

(5 marks)

(e) When a variable is created we need to define a datatype, a name and we need to initialise it. Please explain why each of the 3 elements are required.

(5 marks)

(f) Study the following program carefully and answer the question below:

```
public class PartB{
  static void swap(int a,int b){
         int temp=a;
                 a=b;
                 b=temp;
  }

  public static void main(String[] args){
         int x=5;
                 int y=7;
         swap(x,y);
         System.out.print("x:" + x+ "y:"+ y);
  }
}
```

Write down the output that is displayed when this program is executed and briefly justify your answer.

(5 marks)

(g) Show, with the aid of a clear example, what is meant by the term *scope of a variable declaration*.

(5 marks)

Total (40 marks)

SECTION B – THREE QUESTIONS TO BE ATTEMPTED

QUESTION 2

Consider a staircase of size 4 :

```
   #
  ##
 ###
####
```

Observe that its base and height are both equal to 4, and the image is drawn using # symbols and spaces. *The last line is not preceded by any spaces.*

Write a program that accepts in an integer value from the user N and prints a staircase of size N.

In your answer clearly detail the design of the program as well as creating the code for the solution.

(12 marks for Code)

(8 marks for Design)

Total (20 marks)

QUESTION 3

A Pangram is a sentence constructed by using every letter of the alphabet at least once. Create a program that accepts a string from the user and tells the user if it is a pangram or not. You can assume that the input string is all lowercase characters.

Example : Input = we promptly judged antique ivory buckles for the next prize

Output = This is a pangram

Example: Input = i really hope this example works for me

Output = This is not a pangram

In your answer clearly detail the design of the program as well as creating the code for the solution.

(12 marks for Code)

(8 marks for Design)

Total (20 marks)

QUESTION 4

Create a method that is given an array of integers. The integer values will range from -9 to 9. The method should calculate which fraction of its elements are *positive*, which fraction of its elements are *negative*, and which fraction of its elements are *zeroes*, respectively. Print the decimal value of each fraction on a new line.

-4	3	-9	0	4	1	0	-2	0	7

Positive = 4/10 = 0.4

Negative = 3/10 = 0.333333

Zeroes = 3/10 = = 0.333333

In your answer clearly detail the design of the program as well as creating the code for the solution.

(12 marks for Code)

(8 marks for Design)

Total (20 marks)

QUESTION 5

Create a program that reads from a text file called “dwarves” The file contains the names of the 7 dwarves (Sleepy, Grumpy, Happy, Bashful, Doc, Dopey, Sneezy). Each name is on a new line in the file. There are no extra spaces or tabs on any line. Your program should create a new text file for each dwarf in the source file and write the name of that dwarf into the output file. Each new file should be called after a dwarf.

Example:

Input File

dwarves.txt

```
Sleepy
Grumpy
Happy
Bashful
Doc
Dopey
Sneezy
```

Output files

```
Sleepy.txt      Grumpy.txt          Happy.txt          Bashful.txt
                          D                    zy.tx
Sleepy     Grumpy     Happy     Bashful     Doc     Dopey     Sneezy
```

(12 marks for Code)
(8 marks for Design)
Total (20 marks)