

EDMONTON COUNTY SCHOOL

A LEVEL COMPUTER SCIENCE TRANSITION PACK

"Welcome to Computer Science department"

We look forward to meeting you in September and welcoming you to the Sixth Form.

This Transition Pack contains information to support your transition from GCSE to A Level study.

Please read all the documents ready for the start of term:

1. Welcome to the Computer Science Department
 2. What Independent Learning looks like in Computer Science
 3. Frequently Asked Questions (FAQs)
-
- ✓ **Download the exam board specification** which is available here: <https://filestore.aqa.org.uk/resources/computing/specifications/AQA-7516-7517-SP-2015.PDF> **Read the section called 'Specification at a Glance'**, focusing on the A Level content.
 - ✓ Some of the information will become more useful when you have moved further through the course, such as the A Level specification, so store it where you can revisit it. It will allow you to see what you will be covering over the next two years.
 - ✓ Do not worry if some of the work sounds challenging. A Level work is more difficult than GCSE work after all. Your teachers will be supporting you through this transition. Please talk to us if you are unsure about any aspect of the course.

We look forward to meeting you on **in September 2020**.

Ms Bozkurt – Head of Computer Science

Why study Computer Science?

Learn how to program! The area of Computer Science is an exciting one, with major advances taking place in the development of both hardware and software. You will learn about all aspects of computer science

Course Outline

A Level Computer Science has three units:

Unit 1 - Programming, Data Structures, Algorithms, Theory of Computation (50%)

Unit 2- Data Representation, Computer Systems, Computer Organisation and Architecture, Consequences of Uses of Computing, Communication and Networking, Databases, Big Data, Functional Programming

NEA- Practical project

The course will cover problem-solving and using a computer to help with problem-solving tasks. It will include some advanced electronics, logic circuits, truth tables, systems control, robotics and artificial intelligence, finite state machines, algorithm design, relational databases, systems analysis, data structures and networking. The biggest emphasis will be learning how to write computer software. There will be a large amount of direct hands-on experience, using modern microcomputers together with industry-standard software. Please note that this course does not include learning to use ICT packages such as word processors, desktop publishing and spreadsheets.

Assessment

Students are assessed by two exams at the end of Year 13 worth 80%, plus 20% awarded for a practical project. The first of the exams involves editing a computer program and writing new instructions as part of a practical exam on the computer.

Introduction

You need to be comfortable writing programs before you start the course, as we will begin with object-oriented programming. This sheet should give you plenty of practice if you feel that you will need to improve before you begin studying in September.

We will be using Python in lessons to demonstrate programming concepts as it is the most similar to the AQA pseudocode that you will find in the exams. Paper 1 is written on a computer and involves programming. You can use any language from the following:

- Python
- VB.NET
- C#
- Java
- Pascal

You have been given programming tasks below to help you understand the programming concepts taught at A Level Computer Science. You are required to complete these so that you are prepared for the course when you start in September.

You will need to download Python v3.8.2 from <https://www.python.org/downloads/> , it is free of charge.

Make sure that all programs that you write are easy to understand by another programmer and the end user.

- a. Use meaningful variable names
- b. Add Comments
- c. Program gives clear instructions to the end user what they should do.

Programming Skills for A Level

Assignment

1. Write a program to enter the length, width and depth of a rectangular swimming pool. Calculate the volume of water required to fill the pool and display this volume.
2. Write a program to enter the length and width of a rectangle in cm. Calculate the area and perimeter of the rectangle.

Selection

1. Write a program which will read in a person's age and display a message whether they are old enough to drive or not.
2. Write a program that checks whether a number input is within the range 21 to 29 or not and displays one of the following appropriate messages.
 - Below range
 - Within Range
 - Above Range

Iteration – For

1. Write a program which will print the squares of all the numbers from 1 to 10.
2. n factorial, written $n!$ is the product of all the numbers from 1 up to the number. So $5! = 1 * 2 * 3 * 4 * 5$. Write a program to find the value of $n!$ where n is any positive integer input by the user.
3. Write a program which will ask the user to enter a number of stars per row and the number of rows to be displayed. Then print this grid of stars.
e.g. if they enter 5 and 3, it will print

```
*****  
*****  
*****
```

Hint: you will have to use a FOR loop within a FOR loop.

Iteration – While/Until

1. Using a WHILE loop, write a program that asks the user to enter a number that is greater than 10. If the number is less than 10, display an error message. Loop until a number greater than 10 has been entered. When it has, display a suitable message.
2. Rewrite your program from the previous question but using an UNTIL loop.
Hint: The logic required is the exact inverse of the WHILE loop

String Handling

1. Write a program which asks the user to type in their name and then displays the numbers of letters in their name.

2. Write a program which asks the user to enter their first and last name and then shows their initial followed by a dot and then their last name.
e.g. if their name is John Smith, it displays J. Smith
3. Write a program that will take a sentence from the user as input and automatically convert all characters into lower case, and ensure that the first character is in upper case. The program should check that the sentence ends with a full stop (.) and if it does not, then add one to the end. You can ignore capital letters which should be at the start of proper nouns, and any other type of punctuation.

Arrays

1. Write a program to simulate 30 throws of a dice and count how many 1s, 2s....6s have been thrown

The following pseudocode may help you:

Set up an array to store 6 integers, e.g. Number(0) to Number(5).

Use a for loop to set each number to zero.

Set up a FOR loop to count up to 30

Simulate the throw of a dice by generating a random number
between 1 and 6.

Add 1 to the corresponding element of the array,
e.g. if a 4 is 'thrown' add 1 to Number(4).

End Loop

Using a For Loop Print the number of 1's, 2's, etc.

Procedures (Sub-routines and functions)

1. Write a program that will help the user calculate the volume of various 3D shapes.
First, display a menu asking the user to choose a 3D shape from the following list:

1. Cube
2. Cylinder
3. Sphere

Write separate sub-routines and functions so that when the user selects their option from the menu, they will be asked to input the parameters that will be used to calculate each shape.

E.g. Cube = w^3

Cylinder = $\pi r^2 h$

Sphere = $\frac{4}{3}\pi r^3$

Extra practice

Attempt some of the following programs for extra practice.

Questions taken from AQA Computing (Bond).

1 Max and min

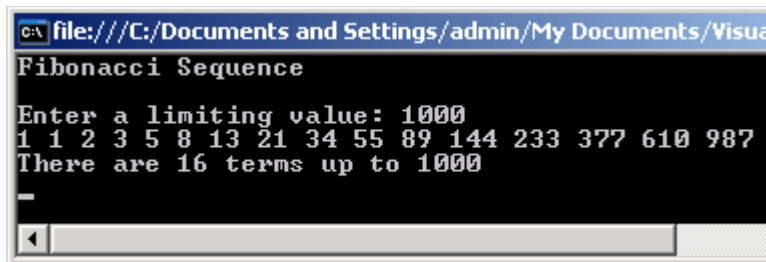
Write a program that inputs a succession of numbers until the rogue value '-1' is entered. The program should then output the highest and lowest of the numbers. Think carefully about what kind of loop to use. One loop should be sufficient to enter the values and identify the highest and lowest.

2 Fibonacci sequence

The Fibonacci sequence of numbers is named after the medieval Italian mathematician Leonardo Fibonacci, or Leonardo of Pisa. It is a pattern often found in nature, for example in the numbers of petals on many flowers.

Starting with two 1s, each number in the sequence is the sum of the two numbers before it, so it goes 1, 1, 2, 3, 5, 8, 13, 21, and so on.

Write a program to input a limiting value, then find how many terms there are in the Fibonacci sequence up to that limit. Your program should also output the actual sequence of numbers, as in the following example:

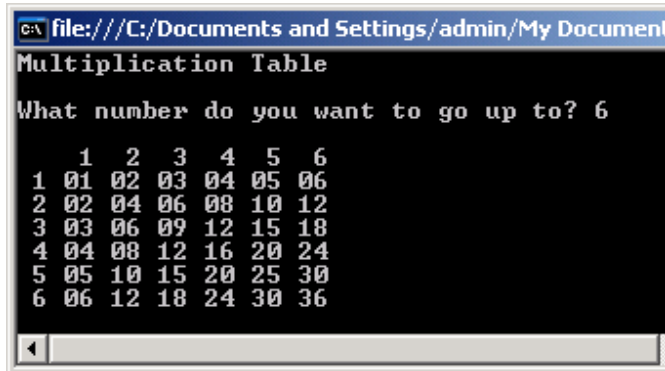


```
C:\ file:///C:/Documents and Settings/admin/My Documents/Visua
Fibonacci Sequence
Enter a limiting value: 1000
1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987
There are 16 terms up to 1000
_
```

Note: You will need variables to fulfil several different roles here. For example, you will need a 'fixed value' variable to hold the limiting value. As you work through the sequence you will need a variable to hold the most recently calculated number in the sequence and a 'follower' variable to hold the preceding number. You will also need a variable to act as a temporary store when moving terms on one place. Finally, you will need a 'stepper' variable to count the number of terms generated.

3 Multiplication table

Write a program to output a multiplication table in which each entry is the product of the numbers in the top row and left-hand column. The user should be able to choose the size of table.



```
file:///C:/Documents and Settings/admin/My Document
Multiplication Table
What number do you want to go up to? 6

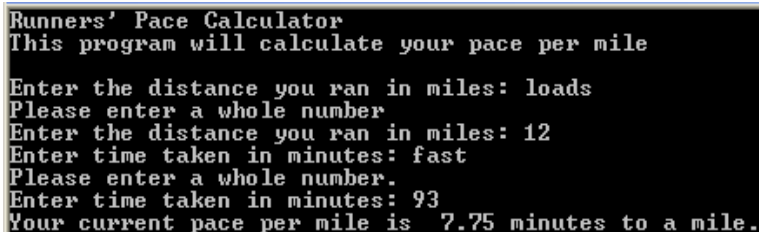
  1  2  3  4  5  6
1 01 02 03 04 05 06
2 02 04 06 08 10 12
3 03 06 09 12 15 18
4 04 08 12 16 20 24
5 05 10 15 20 25 30
6 06 12 18 24 30 36
```

Hint: Use a pair of nested loops for the main part of the table. To format the table neatly you may need to investigate the number formatting features of your particular programming language.

4 Pace calculator

Create a program to calculate the pace per mile for runners at a running club. The program should ask the user to input a distance, then to input the time taken, and will calculate the pace.

The program should use exception handling to trap errors in the input, for example, entering a real number or string instead of an integer. The program should loop until acceptable input is provided.



```
Runners' Pace Calculator
This program will calculate your pace per mile

Enter the distance you ran in miles: loads
Please enter a whole number
Enter the distance you ran in miles: 12
Enter time taken in minutes: fast
Please enter a whole number.
Enter time taken in minutes: 93
Your current pace per mile is 7.75 minutes to a mile.
```

Exam Style Question 1

Guessing Game

Write a two-player guessing game.

The game should first ask Player One to choose a whole number between 1 and 10 (inclusive). Player Two should then be asked to guess the number chosen by Player One. Player Two gets up to five attempts to guess the number.

Player Two wins the game if they correctly guess the number and the computer should display the text "Player Two wins".

However, if Player Two is unable to guess correctly after 5 tries, Player One wins the game and the text "Player One wins" is displayed.

Identifier	Data Type	Purpose
NumberToGuess	Integer	Stores the number entered by Player One
NumberOfGuesses	Integer	Stores the number of guesses that Player Two has made so far
Guess	Integer	Stores the most recent guess made by Player Two

What you need to do

Write the program and include variables from **Table 1** plus any other that you feel are needed.

Test the program by conducting the tests **Test 1** and **Test 2**.

Test 1

Test that your program works correctly by conducting the following test:

- Player One enters the number 0
- Player One enters the number 11
- Player One enters the number 5
- Player Two enters a guess of 5

Test 2

Test that your program works correctly by conducting the following test:

- Player One enters the number 6
- Player Two enters guesses of 1, 3, 5, 7, 10

Exam Style Question 2

Run-Length Encoding

One method that can be used to compress text data is run length encoding (RLE). When RLE is used the compressed data can be represented as a set of character/frequency pairs. When the same character appears in consecutive locations in the original text it is replaced in the compressed text by a single instance of the character followed by a number indicating the number of consecutive instances of that character. Single instances of a character are represented by the character followed by the number 1.

The examples below show how text would be compressed using this method.

Original text:

AAARRRRGGGHH

Compressed text:

A 3 R 4 G 3 H 2

Original text:

CUTLASSES

Compressed text:

C 1 U 1 T 1 L 1 A 1 S 2 E 1 S 1

What you need to do

Write a program that will perform the compression process described above. The program should display a suitable prompt asking the user to input the text to compress and then output the compressed text.

Test 1

Test the program works by entering the text:

- AAARRRRGGGHH

Test 2

Test the program works by entering the text:

- A

Other Projects

The following can extend your understanding the subject before you start, and are worth also looking at:

Theory

You must watch videos on Craig and Dave website to make sure you understand theory for all the topics in this specification. Some topics are overly complex and requires A Level Maths skills and therefore you are strongly advised to watch these videos to improve your understanding.

<https://student.craigndave.org/aqa-alevel-videos>

Maths

It is helpful if you can describe and identify numbers from the following sets:

- Integers
- Real numbers
- Rational numbers
- Irrational numbers

Object Oriented Programming

The best A Level coursework projects are written using Object Oriented Programming. This can be a challenging concept to follow at first, but it might be helpful if you have looked at some of the following resources. Do not worry if you do not understand this though – You will learn it!

Sign up to FutureLearn courses and complete courses online to improve your programming skills

<https://www.futurelearn.com/subjects/it-and-computer-science-courses/coding-programming>

Other Programming

Although it is not recommended to use Unity for your coursework, you could have a look at it, especially if you hold ambitions to enter the gaming industry.

The best set of resources which are free are on the Brackeys YouTube channel.

https://www.youtube.com/channel/UCYbK_tjZ2OrIZFBvU6CCMiA

A Level Computer Science FAQs

How much of the course is theory? How much is practical, writing programs?

Over the whole course, approximately half and half, but there is a lot of practical work to do when learning how to write programs/software. You can expect to spend almost all of the first term of Year 12 up to Christmas doing practical work.

What is the weighting of the above?

Approximately 50% each. In Year 13 the practical coursework makes up 20% of the total A Level. You will also sit a practical on-screen examination in which your programming skills will be assessed.

Is there a lot of Maths involved? Do I need to take Maths A Level as well?

Taking Maths at A Level will certainly help with Computer Science and is advised, but it is not compulsory. The Computer Science syllabus contains some Maths when covering number bases (e.g. binary, hexadecimal) and in Boolean algebra.

What is the difference between ICT and Computer Science?

Put in simple terms, ICT is concerned with learning to use software written by others (e.g. databases, spreadsheets, etc.) whereas Computer Science is concerned with problem solving and writing algorithms to solve these problems which can then be converted into programs in a programming language.

What other subjects fit well with Computer Science?

Maths and physics in particular

What will I expect to be able to program by the end of Year 12? Year 13?

By the end of Year 12 you will have learned all the basic building blocks of coding structured software such as assignment, iteration, selection, subroutines, file handling, etc. By the end of Year 13 you will be a very competent programmer and have experience of developing object-oriented programs and graphical user interfaces. In year 13 you will also cover functional programming and SQL, whilst many of our students also learn another language such as Java or Python.

Do we write games software?

Yes, and depending on your choice of coursework in Year 13, you may write quite a lot of this!

Do I get much homework?

In line with all other A Level subjects at ECS, we would expect you to do weekly homework.

Is there any coursework?

There is a practical examination set by the exam board each year which involves modifying

and adding extra functionality to the skeleton program code provided by the exam board. These are usually very interesting problems. In year 13, the non-exam assessment is worth 20% of the A Level, and involves students writing a substantial piece of software.

What jobs would I be able to apply for with Computer Science A Level? What courses could I take at university which link with Computer Science?

You may end up becoming a programmer, systems analyst, IT technician, etc. Courses at university would include Computer Science (some universities require A Level Maths for this degree), Games Programming, Systems Design, etc. Even if you don't pursue this subject any further after your A Levels, the problem-solving skills you will acquire will be extremely useful in other subject areas. With experience, salaries for some of these jobs may exceed £100,000 per annum. A senior network manager in a large school may earn up to £50,000 a year, whereas a senior systems analyst working on a substantial project may earn well over £100,000 per annum.