Welcome to A level Computer Science at Macmillan Academy Post 16. We are delighted that you are considering this subject and hope that you will find completing this bridging project both enjoyable and challenging. We would like you to complete as much of it as you can and bring this work along when you start your new course with us.

At A level, you will study two examined units and complete a substantial programming project in Y13. If you have studied Computer Science at GCSE then you will be familiar with many of the topics, though you will go on to study them in significantly more depth. If you haven't studied the subject at GCSE, then hopefully this will give you an idea of the topics you will be studying and allow you to make a more informed choice about your journey through P16.

Throughout this transition project, you may find the following resources useful:

Teach-ICT website

website: http://www.teach-ict.com/

Username: ha52lx

Password: computing

Python

Website: http://www.python.org

Download the latest version (3.8.3 for Windows/Mac/Linux)

COMP 1 – Section 1 – The development of the computer

Students of computer science should have a good working knowledge of this history of computing. It's important to know your subject – not just what it is now and where it is going, but where it came from and how we got here. We learn lessons from the past and use them to inform the future.

For each of the tasks below, create some notes detailing your research and findings. You will need quite a bit of detail before you can complete the final task in this section. You will be expected to hand in your notes and completed task when you start!

Task 1 – CPU Research

The processor or CPU is the most important part of the computer system. We talk about it being the "brain" of the system, but this isn't really correct. A brain has the capacity for independent thought and abstract thought, whereas a CPU is just an electronic machine which only follows rules. The first CPU as we know it was produced by Intel in 1971 and was called the 4004. It was constructed from 2300 transistors, whereas a modern Intel i7 processor has around 2.3 billion. It ran at 740KHz compared to modern processors which operate in the region of 3.5GHz.

Investigate beginning with the 4004 and including the major processors which came around. You are not expected to include every processor ever made, but there are some significant milestones which were achieved along the way, for example duel cores, reaching the different speed boundaries (MHz, GHz etc), different manufacturers, different architectures. You will need to do some considerable research to find out the key events for this timeline. Much development happened in the early 1980's with the advent of home computing. Prior to that, computers had been massive in both size and cost, but Sir Clive Sinclair made some significant breakthroughs in chip design resulting in cheap, and small home computers. What is Moore's law? Does it still apply?

Task 2 – Computer Research

Now you have an understanding of the development of the CPU, and you have probably looked at the 1980's as a point of massive growth, it's time to look at how you got to be using the computer you are now sat in front of. You should investigate the development of the desktop computer, from it's very early origins as a business machine in the 1970's, through the boom of the 1980's, the standardisation of the 90's and the monopolisation of Apple and Microsoft which followed. There was also the split and then subsequent development of the games console, which is simply a computer with a different interface and appearance. From humble beginnings with dozens of different companies to the PC and Mac – how have we got here? A good place to start is this website: https://old-computers.com/museum/default.asp

Task 3 – Key Player Research

CPUs, computers and now people. Who are the key players in the development of this field? Who are the people responsible for the technology which is now so much a part of your life? Growing up the 1970's and 80's would be so different – no home computers, no satellite TV, no mobile phones. Some key players came along and challenged all of that – you'll be

surprised by some of the them – you won't associate them with how they "fired" the development of the home computer!

Task 4 – Software Research

Computers don't work without software – early home computers had operating systems built in – they booted up to a flashing cursor and you had to write your own programs. Today out computer boots up into an elaborate graphical user interface with a mouse available for navigation, and even touch screen. How did we get here from a cursor/prompt to graphical displays? How were they developed, when and what did they look like? Games are often considered the driving force for development – the need for better and better performance. Some games were massive gamechangers (Space Invaders, Manic Miner, Doom come to mind). Look into software and how it influenced the development of the computer.

Task 5 – Create

Create a timeline of the development of the computer. Decide what the key events are that need to go in there – some were responsible for enormous changes in the timeline, whereas others were much more subtle.

Success Criteria

- Your timeline should cover at least 30 key events spanning the last 50 years as a minimum (you will need to include many more than this to have any level of detail)
- It should include CPU, Computers, people and software (perhaps use some sort of logo/symbol/key for the type of entry.
- Each event should have its significance explained
- Each event should be dated

COMP 2 – Computational Thinking

The coding challenges below will let you check your skills. Part of the transition to A-level is combining skills, and also ensuring that you plan and test your work thoroughly, so think about how you can re-use components and design your code for readability and robustness.

1. Write a program to:

a. Ask the user to input

i. Their first name

ii. Their surname

iii. A date, in the format DD/MM/YYYY

b. The program should then output a customer ID as follows:

i. The date in the format YYYYMMDD, then the first three letters of the surname, then the first initial, then the length of their first name. All letters should be in capitals

ii. For example, John Smith, 27/05/2017 would give 20170527SMITHJ4

c. The program should validate any inputs and keep asking for inputs until the user enters correct details or types “quit” at any point

- Plan your algorithm first, using a flowchart or pseudocode
- Code your algorithm, and provide evidence of both your code and the working output
- Create a test plan for your algorithm, including testing your validation with normal, boundary and erroneous data

2. Write a program to:

a. Ask the user to input

i. The name of a product

ii. Its cost in pounds

iii. The program should keep asking for inputs until the user types “None”

b. The program should then output:

i. The name and price of the most expensive item

ii. The name and price of the least expensive item

iii. The average price of the items

iv. The total cost of the items

- Items over £50 get a 5% discount
- VAT is added at the end at 20%
- The program should validate any inputs
- Plan your algorithm first, using a flowchart or pseudocode
- Code your algorithm, and provide evidence of both your code and the working output
- Create a test plan for your algorithm, including testing your validation with normal, boundary and erroneous data.

COMP 2 – Problem Solving

The following puzzles will help you to develop you logical thinking skills. There are many good books of puzzles, plus countless online sources to test your skills. Some recommendations are given later.

The following puzzles are representative of classical problems and problem solving strategies. You can solve each one by trial and error, but you encouraged to think about the strategy you employed to solve the problem. Note that there are discussions of each problem available online if you want to investigate them in more detail.

Two good general strategies to try are:

- Can you solve a simpler version of the problem first?
- Can you draw a diagram to help you visualise the problem?

After that, you have your standard computer science strategies:

Decomposition

- Can you split the problem down into smaller parts to solve?

Abstraction

- Can you remove any unnecessary details to focus in on only what you need to solve the problem?
- Be careful – are you sure that you have kept the right information?

Generalisation and problem recognition

- Is this puzzle a specific example of a problem for which there is a general solution? If so, how does it apply in this case?
- Do you recognise the problem from somewhere else, or is it similar to something else?
- You may need to generalise the problem to identify the core features so that you can spot equivalent problems.

Another important strategy is to ensure that the problem is well-defined. This means that you know:

- The goal: what you are trying to achieve
- The givens: what you know at the start, or your starting conditions
- The resources: what you have available to solve the problem
- The constraints: any rules that limit your solution
- The ownership: who or what is carrying out each part of the solution

Sometimes just working through the problem definition carefully is enough to give the required insight.

The complete work on problem solving is Polya's "How To Solve It"; there are many sources for this online if you are interested.

The problems

The Princess in the Castle

A princess lives in a long corridor in a castle. The corridor has 17 rooms, numbered 1 to 17 inclusive.

Each night the princess sleeps in a different room according to the following rules:

- On the first night of the year she sleeps in a random room
- Each night she moves to an adjacent room; she never sleeps in the same room on two nights in a row and she always moves exactly one room left or right along the corridor
    - For example, if she is currently sleeping in room 12, then on the next night she will
    - either be in room 11 or in room 13
    - If she is in room 1, then she must be in room 2 on the next night as she cannot move
    - in any other direction (the same is true for room 17 – she must move to room 16 next)

A prince wishes to marry the princess. To do this he must find her room in the castle. However, whenever he sneaks into the castle at night, the guards quickly find him and throw him out!

Therefore he only has time to search one room each night.

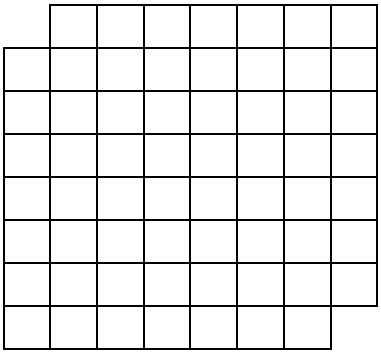
The princess is unable to give the prince any clues to her location, and the prince has no knowledge of her location, other than whether or no she was in the room he last tried.

What strategy should the prince follow in order to find the princess in a finite time?

What is the maximum number of nights the prince needs to search before he can guarantee finding the princess?

Covering a Chess Board

Imagine a standard 8x8 chess board. Now cut off two diagonally opposite corners squares to get a shape like this:



I also have a pack of dominoes. Each domino is exactly the right size to cover two squares on the chess board, either horizontally or vertically. (The dominoes cannot be placed diagonally.)

Is it possible to cover the board with dominoes so that each domino covers exactly two squares, with no overlaps and without any dominoes "hanging off" the edge of the board? If so, how do you do it?

If not, why not?

Hat, hat, hat…

I have taken a group of students on a school trip. I want to organise them into two groups and so I have given each one a coloured hat. Some hats are red, while others are blue. Each child can see everyone else's hats, but not their own.

I am feeling obtuse, so I have asked the students to get themselves into two groups based on the colours of their hats, with all the red hats together and all the blue hats together. But! I have told them they are not allowed to talk or communicate in any way.

What strategy should they use to form the two groups?