Translators and facilities of languages

High Level Language

Most programmers write programs using a high-level programming language. These languages closely match spoken and written language and use keywords to represents commands such as print, input and if etc.

```
num1 = int(input("Enter num1: "))
num2 = int(input("Enter num2: "))
total = num1 + num2
print(total)
```

Low level languages

Machine Code

The commands that tell a computer what to do are written in binary, known as machine code. Each command is usually represented using hex and represents a different instruction.

```
C000: A0 00
C002: B9 00 C1
C005: 20 6F D2
```

Assembly Language

Assembly language sits between machine code and high-level languages and uses abbreviations to represent the commands rather than the hex that machine code does.

```
INP
STA 06
LDA A1
```

Translator

Any program written in a high-level language is known as source code. Source code must be translated into machine code before the computer can understand and execute it.

Compiler

A compiler takes the source code as a whole and translates it into machine code all in one go. Once converted, the object code can be run multiple times without further translating, unless the original source code is altered in which the whole thing will need to be translated again.

Interpreter

An interpreter translates source code into machine code one instruction at a time and runs it immediately rather than translating all the code before it is run.

Assembler

Assemblers translate assembly language code into machine code instruction one line at a time.

INTEGRATED DEVELOPMENT ENVIRONMENT (IDE)

Editors

Editors are software which allow programmers to write and edit code. Editors are often fairly simple but usually offer facilities such as:

- Colour coding
- Autocorrect
- Autocomplete
- Automatic indents

These tools help to improve the readability of code. However, they do not usually help to identify errors.

Error Diagnostics

Error diagnostics is commonly known as “debugging” and help programmers to locate and fix errors:

- Breakpoints enable a program to be paused or stopped at predetermined points, to allow them to test out small sections of the program
- Variable tracing lets the programmer see the values stored in variables as the program is running which helps them visualise what is happening
- Some IDEs highlight exactly where a syntax error occurs

Runtime Environment

A runtime environment (RTE) is special software that allows a program to run on a computer even if it is not designed to run on it. It works on different platforms, meaning a program can be developed in a Windows environment which is intended for a MAC.

Translators

Many IDEs include a translator to allow the programmer to test the program and make small alterations before the final program is compiled into an executable file.

Translators and facilities of languages

Revise it

Read through the handout and then select a revision technique from those described in this section, you can even do more than one if you want!

Highlight

Highlight key words (maximum of 2 per sentence) and then cover the page and try to write down all the key words you can remember. Go back and fill in all the ones you have missed.

Mind map

Using the handout, draw a mind map and include as many colours, images and diagrams as you can to illustrate it

Bullet Points

Write the main headings (leaving space between each) and then write bullet points of the main key points you need to remember under each heading. Re-read the handout and add any missed points to your list.

Post-it notes

Write a key word and the definition on a post-it note and stick them around your study area as a reminder of the terminology.

Record your notes

Re-write the handout in your own words and record yourself using your phone as you read your notes aloud.

TEST YOURSELF

Cover your notes and the answer before you attempt to answer this practice exam question.

Outline TWO differences in the way a compiler and an interpreter would translate a program written in a high-level language [4 marks]

Mark your answer

Give one mark for each part of an explanation which compares a compiler and an interpreter (maximum four marks)

- A compiler translates all the code at once before the program can be run *[1 mark]* but an interpreter would translate the code line by line . *[1 mark]*
- A compiler would create a list of errors at the end of the translation *[1 mark]* but an interpreter would find the first error and then stop translating. *[1 mark]*
- A compiler would produce an executable program [*1 mark]* but an interpreter would not. *[1 mark]*