# Part 1

## Question 1)

|  |  |
| --- | --- |
| C | Python |
| Low-level | High-Level |
| Manual memory management | Manages memory automatically |
| Faster and more efficient | Slower and less efficient |
| More complex syntax | Easy-to-read syntax |
| Structed oriented | Object oriented |

|  |  |
| --- | --- |
| PROS | CONS |
| Better performance | Complex syntax |
| Low-level control over memory | Memory leaks and buffer overflows |
| Used in embedded systems | Limited standart library |
| High speed | Absence of error handling |
| Support the pointers | No strict type and run - time checking |

## Question 2)

|  |  |
| --- | --- |
| Int: | Integers, 4 byte. |
| Char: | A single character, 1 byte. |
| Float: | Single-precision floating-point numbers, 4 byte. |
| Double: | Double-precision floating-point numbers, 8 byte. |
| Long: | Extended range integer values, 4 byte. |
| Short: | Short-range integer values, 2 byte. |
| Bool: | True or False, 1 byte. |

## Question 3)

A pointer is a derived data type. It can store the address of other variables or represent memory location. The type of a pointer represents the type of data it points to. Void\* means a memory address without specifying the data type it points to.

## Question 4)

In Code 1, "&number" gives us a float pointer. The float pointer is converted into an int pointer. Then, the value is taken by dereferencing. Shortly, float memory is treated as int memory. Code 1 gives a different result from the original value.

In Code 2, the program typecasts the float variable number into an integer, giving directly -42, which is the integer part of the float number.

# Part 2

## Question 1)

A black screen with white text

Description automatically generated

## Question 2)

A black screen with white text

Description automatically generated

## Question 3)

A screen shot of a computer

Description automatically generated