IB Questionnaire - Computer Science

Assessed by: Michal Špano

Collection of frequently asked questions of the IB programme in Computer Science

---

Approaches to learning

Every IB course should contribute to the development of students' approaches to learning skills. As an example of how you would do this, choose one topic from your outline that would allow your students to specifically develop one or more of these skill categories (thinking, communication, social, self-management or research).

Topic: Online platform for effective learning

Contribution to the development of students' approaches to learning skills (including one or more skill category):

Specified skill categories shall be easily enhanced by creating an online-based platform for students that will ensure that they can effectively use the skills they have acquired and put the obtained knowledge into practice. With that in mind, the students could engage in conversations with like-minded people and share their knowledge with others. As a result, the students will be able to use the skills they have acquired to solve problems in the real world, manage their time and improve their capabilities.

State the fundamental operations of a computer.

The following list of operations is a typical example of a computer's basic operations. The list is as follows: **inputting**, **processing**, **outputting**, **storing** and **retrieving data**, also referred to as *controlling* the computer to a certain extent, and **controlling the computer's memory**.

Distinguish between fundamental and compound operations of a computer.

Compound operations of a computer refer to an operation that involves a variety of other `sub-operations` and/or `sub-processes` or stages of other **operations**. For instance, the operation of a computer to process data of a **specific computer program** is a compound operation, as it involves several sub-operations. On the other hand, **fundamental operations** don't depend on other operations, and are *independent* of other operations.

Explain the essential features of a computer language.

To define some of the **key features** of a `programming language`, one must understand that programming languages as a whole can be differentiated based on various features. Some of the most common types of programming languages are:

- **Object-oriented programming languages** (`OOP`), which are based on the concept of **objects**, such as `Java`, `C++` or `Objective-C`.
- **Concatenative programming languages**, which are based on the concept of **concatenation**, such as `Forth`.
- **Functional programming languages**, which are based on the concept of **functions**, such as `Haskell`, `Scala` or `Elixir`.

Still, a well-defined **programing language** must include some of these features (regardless of the type of programming language):

1. **Syntax**: The syntax of a programming language is the way in which the language is written i.e., denoted using a set of rules and special characters. Modern languages focus on **user-friendly syntax**, which is easy to understand and use, such as `Ruby` or `Python`.

2. **Compilation**: The compilation of a programming language is the process of converting a program written in a programming language into a machine-readable form. This `process` varies based on the language type, though it is usually done by a compiler that ought to be effective and memory-efficient.

3. **Package management**: The `package management` of a programming language is the process of organizing the files of a program into a `package` that can be easily deployed, shared and used. Languages like `Rust` or `Go` have a `package management` system that allows for easy deployment and sharing of programs, using the `cargo` and `go` packages respectively.

Notes: The `package management` of a programming language is not the same as the `package management` of a computer operating system. For languages without a `package management` system, alternatives such as `Docker`, `NPM`, `Git` or `Kubernetes` can be considered.

Explain the need for higher level languages.

Higher programming languages, just like `Python` or `C#` to name a few, are a great way to build **powerful** applications and programs with enhanced readability and maintainability. However, the question of **performance** arises, as the performance of a higher level language is often much lower than that of a lower level language. So, should you sacrifice the readability of a program for performance and vice versa? That's entirely up to the **developer** as few languages are designed to be used in a specific context, hence called **general purpose** languages.

Outline the need for a translation process from a higher level language to machine executable code.

Virtually, computer languages are **translated** into machine executable code. This process is called **translation**. The translation process is usually done by a **compiler**, which is a program that translates the source code into machine executable code. Programming languages are just a single instance how to *make a computer do something*. Regardless of the type, `computers` are only able to *communicate* using a specific set of instructions, those include a properly defined machine language, such as `assembly language` or `machine code`. In theory, large blocks of code are broken down into smaller pieces, each of which is translated into a single instruction in the form of a `bytecode` or a `binary sequence` of bits.

For instance, consider the following case of the `C` programming language, a low-level language with a great emphasis on the `memory` and `control flow` features.

1. Preprocessing
2. Compilation
3. Assembly
4. Linking

*More can be obtained here

Define the operators =, ≠, <=, >, >=, mod, div

In **Computer Science**, jus like in **mathematics**, the operators are used to compare values. The following table lists the operators and their meanings:

Operator	Meaning
`=`	equal to
`≠`	not equal to
`≤`	less than or equal to
`<`	less than
`>`	greater than
`≥`	greater than or equal to
`mod`	modulo operand
`div`	division

Notes: The `mod` operator is used to find the **remainder** of a division. The `div` operator is used to find the **quotient** of a division.

Analyse the use of variables, constants and operators in algorithms.

To define a well-performing algorithm, one must understand the **use of variables**, **constants** and **operators**.

1. **Variables** are used to store values.
2. **Constants** are used to store values that are not changed during the execution of the algorithm.
3. **Operators** are required to compose the algorithm, i.e. to establish logical relationships between the values stored in the variables and the constants.

Consider the following example of an algorithm:

```
int main(void) {
    int a = 0; // variable
    const int b = 10; // constant
```

```
    // provisional algorithm
    for (int i = 0; i < 10; i++) {
        a = a + b;
    }

    <...>

    /*
    * The value of the constant `b` is not changed during the execution of
the algorithm.
    * While the value of the variable `a` is changed during the execution
of the algorithm.
    * Editing the value of a `constant` will result in an error.
    */
}
```

Construct algorithms using loops, branching.

Consider my own implementation of the `bubble sort` algorithm in `Python` that demonstrates the use of loops and branching.

```
def selection_sort(l: list):
    for i in range(len(l)):
        min_index: int = i  # Keep track of the local minimal index in the
sub-list
        for j in range(i + 1, len(l)):
            if l[j] < l[min_index]:
                min_index = j

        # Swap the min. detected value with the current index
        l[i], l[min_index] = l[min_index], l[i]
    return l

# IO test
print(selection_sort([1, 2, 3, 4, 5, 6, 7, 8, 9, 0, -1]))
```

Expected behavior:

```
[-1, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Describe the characteristics and applications of a collection.

In programming, a **collection** is an `object` used to represents instances of data that share a common characteristic. A **collection** has an underlying **structure**, which is the way in which the data is represented. Consider the following example of a **linked list** in `C`:

```c
struct node {
    int value; // store an int value
    struct node *next; // store a 'pointer' to the next node
};
```

Notes: The `struct` keyword is used to define a new type of data structure in `C`. Other well-used data structures include `arrays`, `lists`, `trees` and `graphs`.

Construct algorithms using the access methods of a collection.

Build-up of the `linked list`, where the **access method** refers to the `pointer` to the next node. Consider the following example of a `linked list` in `C`:

```c
struct node {
    int value; // store an int value
    struct node *next; // store a 'pointer' to the next node
};

int main(void) {
    struct node *head = NULL; // initialize the head of the list
    struct node *current = NULL; // initialize the current node

    /*
    * Assign values from 1 through 10 to the list
    * Note: the `head` is the first node of the list
    */

    for (int i = 1; i <= 10; i++) {
        struct node *new_node = malloc(sizeof(struct node)); // allocate
memory for a new node
        new_node->value = i; // assign the value to the new node
        new_node->next = NULL; // initialize the next node to NULL

        // If the list is empty, the new node is the head
        if (head == NULL) {
            head = new_node;
        } else {
            current->next = new_node; // assign the new node to the next
node of the current node
        }
        current = new_node; // assign the new node to the current node
    }

    /*
    * Accessing values of the list
    */

    current = head; // start from the head
    while (current != NULL) {
```

```
            int val = current->value; // get the value of the current node
            current = current->next; // move to the next node
        }
    }
```

Notes: The `malloc` function allocates memory for a new node. The `free` function frees the memory allocated for a node. The `->` operator is used to access the value of a node.

Discuss the need for sub-programmes and collections within programmed solutions.

Well-designed data structures, simply collections, represent a more *complex* way of storing data. In the context of programming, a **sub-programme** is a set of instructions that can be executed independently of the main programme. A **collection** is a set of data that share a common characteristic. By implementing such a data structure, the programmer can access enormous amounts of data in a more efficient way rather than using the conventional way of storing data in a list.

Every IB course should contribute to the development of international-mindedness in students. As an example of how you would do this, choose one topic from your outline that would allow your students to analyse it from different cultural perspectives. Briefly explain the reason for your choice and what resources you will use to achieve this goal.

Topic: International mindedness

Contribution to the development of international mindedness(including resources you will use):

International-mindedness is a concept that is used to describe the way in which a person thinks, behaves and speaks. It is a way of thinking that is based on the idea that the world is a **global community**. With that in mind, it is important to understand how people think and behave in different cultures. Only then we can apply the gathered information to extend our understanding of the world. I'd personally opt to a method that would convey the idea of a **global community** in a local setting, namely in a classroom or a lab. To demonstrate such proposal using a concrete example from the outline, one could consider the topics of **networks** and **internet-based communication**. **Standards** as means of `protocols` or other elements regarding internet-based communications are an underlying mechanism of the internet. Likewise, such **standards** are involved in every-day human communication and can vary from a culture to another. Similarly, in order to communicate efficiently, one should use the expected **standards** of a culture and thus enhance one's involvement in forming a more equitable and effective society from a variety of ethnic perspectives.

Through the course it is also expected that students will develop the attributes of the IB learner profile. As an example of how you would do this, choose one topic from your course outline and explain how the contents and related skills would pursue the development of any attribute(s) of the IB learner profile that you will identify.

Topic: Development of the IB learner profile

Contribution to the development of the attribute(s) of the IB learner profile:

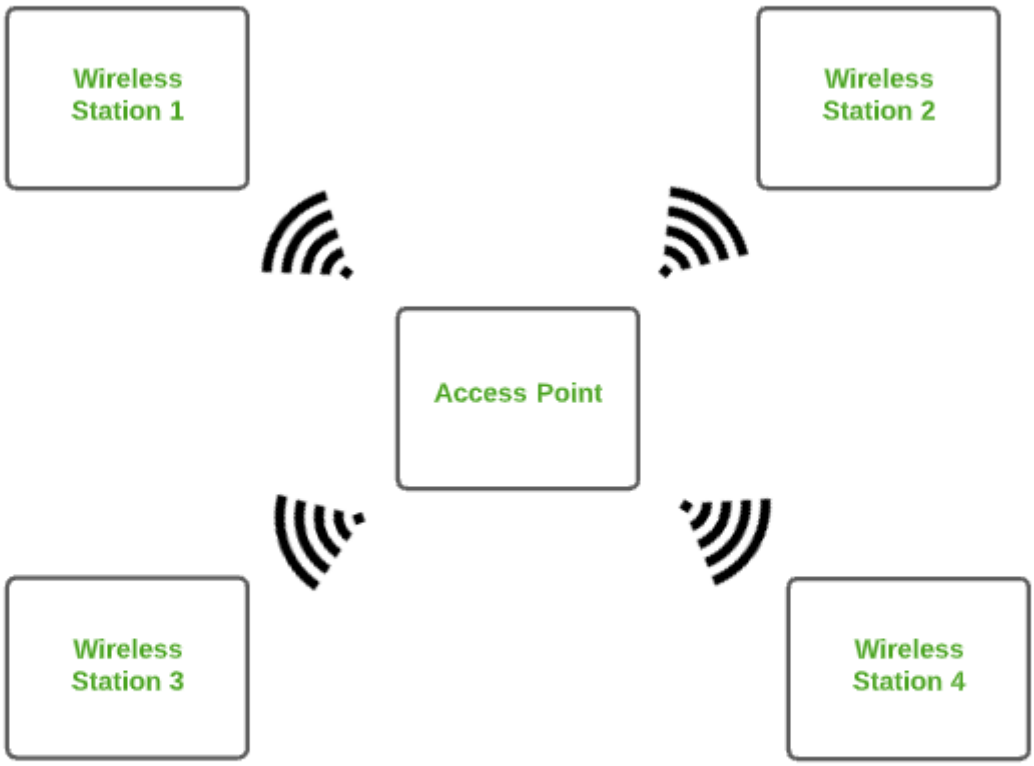
In order to strengthen one's attributes as means of developing the IB learner profile, it is important to understand the way in which the IB learner profile is developed. The IB learner profile is a set of attributes that describe a person's knowledge, skills and abilities. The IB learner profile is developed through a series of activities that are designed to develop the attributes of the IB learner profile. Likewise, students are well encouraged to exchange knowledge and skills through engaging conversation or taking place in projects with like-minded peers. Just like in programming, us, humans, operate on various levels - that be *lower* level or *higher* level tasks. Those imply the need of a certain knowledge of the **lower level** criteria in order to be able to perform the **higher level** tasks accordingly. Virtually, every human being is a **higher level** collection of tasks built upon **lower level** intrinsic tasks. That being said, one's understanding of the fundamentals of a certain field of science is a significant factor in the development of personal skills, knowledge and abilities even regarding an **IB learner profile**.

Identify different types of networks

Different types of **networks** (with their respective *abbreviations*) include:

Abbreviation	Network
PAN	Personal Area Network
LAN	Local Area Network
WAN	Wide Area Network
WLAN	Wireless Local Area Network
CAN	Campus Area Network
MAN	Metropolitan Area Network
SAN	System Area Network
EPN	Enterprise Private Network
VPN	Virtual Private Network
HAN	Home Area Network

A very common application of `WLAN` is `Wi-Fi`:



Outline the importance of standards in the construction of networks.

Generally speaking, companies and other business enterprises wouldn't be able to communicate with each other unless they follow a certain set of rules. These rules are collectively called **standards**.

Identify the technologies required to provide a VPN service.

ATM is a network that provides a secure connection between two computers. The connection is established using a **VPN** service. Moreover, **MPLS** is a protocol that is used to provide a secure connection between two computers.

Evaluate the use of a VPN.

VPN is a **virtual private network**. It is a network that is not directly connected to the internet. Instead, it is connected to a **VPN** service. Hence, the use of a VPN enables the user to interact with the internet through a secure connection oftentimes required for accessing sensitive information or data.

Define the terms: protocol, data packet.

1. `Protocol`: A **protocol** is a set of rules that are used to define the way in which a computer communicates with other computers. Example include: `IMAP`, `POP`, `SMTP` or `HTTP`. `HTTP`, briefly, is a protocol that is used to transfer data over the internet.
2. `Data packet`: A **data packet** is a unified end-user data unit. It is a set of data that is transferred over the internet, abbreviated as `API` - **Application Program Interface**.

Explain why protocols are necessary.

In modern **data communication**, protocols are `standards` that define the set of rules required for data to be transferred over a **communication medium** - the 'internet'. In other words, the rules that are used to define the way in which data is transferred over the internet are called **protocols**.

Explain why the speed of data transmission across a network can vary.

On the whole, **data transmission** can be affected by a variety of factors that include: geological location, traffic patterns, network congestion, and other factors. Also, the speed of data transmission can vary depending on the type of network.

Explain why compression of data is often necessary when transmitting across a network.

Data compression is genuinely used in the field of data transmission. It is a process in which data is reduced in size by removing unnecessary information. Various **compressing methods** are applied in instances of large data sets to increase the speed of data transmission.

Outline the characteristics of different transmission media.

A **transmission medium** describes the physical boundary between the transmitter and the receiver. These can be broken down into two types:

1. **Guided media** (also known as wired or bonded)
2. **Unguided media** (also known as wireless or unbounded)

- The features of **guided media** are: High speed, security, reliability, and use for shorter distances
- The features of **unguided media** are: Less secure, used for greater distances, the signals are represented through dense signals of waves transmitted and/or broadcasted through the air.

The most common examples of **guided media** are:

1. Twisted Pair Cable
2. Coaxial Cable
3. Optical Fiber Cable
4. Stripline

The most common examples of **unguided media** are:

1. Radiowaves
2. Microwaves

3. Infrared
4. Bluetooth
