Karla N Robles

1. Explain how compiled, hybrid, and interpreted programming languages are different in terms of their implementation. When would you consider using a compiled programming language over an interpreted one?

Compiled programs (e.g. C++) are translated into machine language run directly on CPU hardware. Compilation and execution are OS specific. The source program goes through a lexical analyzer, syntax analyzer, intermediate code generator, semantic analyzer, optimization, & code generation before running on the CPU

Interpreted programs (e.g. PHP) are interpreted by another program known as an interpreter. The source code is not translated into machine language.

Hybrid programs (e.g. Python) are a compromise between compilers and pure interpreters. They are executed on virtual machine (software) instead of CPU. Here, a high-level language is translated to an intermediate language that allows easy interpretation.

A compiled program should probably be used in large or commercial applications where efficiency and speed are important.

2. Show an example how the type checking feature can be helpful to improve reliability of a programming language.

Consider a function that accepts a char argument: `char-func(char n)` in a C-based language. If the function is called `Char_func(5)`, an error will be thrown to alert the programmer this data type (int) is not compatible. The program language's reliability is improved with type-checking because it helps to diagnose type-associated mistakes.

3. Explain with an example how the support for abstraction can improve writability of a programming language.

Abstraction refers to the ability to define & use user-defined data-types and methods, thereby hiding some/all of the steps in an operation. This can improve writability by improving the ease with which the code is implemented.

4. In your preferred programming language identify and implement the following data types. In case your preferred programming language does not directly support a data type, show how existing types and constructs in your programming language can be used to obtain an equivalent data representation

Programming language: C++

a. Integer type
b. String type
c. Character type
d. Floating type
e. Boolean type
f. Arrays - one-dimensional and two-dimensional

g. Lists as heterogeneous array
h. Enumeration type
i. Associative arrays
j. Record types

```
#include <string>
int main () {
    int some_int; //OR int some_int = 5;
    string str;
    char chr = 'a';
    float flt = 3.14;
    bool condition = false;
    int oneD[2] = {1, 2};
    int twoD[2,2] = {{1,1}, {2,2}};
    // user-defined, no such data type
    enum pet (Dog, Cat, Hamster);
    map < key, val> aarray {{"key1", 1}, {"key2", 2}};
    struct aRecord
    {
        int recordVal1;
        Char recordVal2;
    }
}
```

5. Python programming language can create an iterator object of enumeration type as below. Show an equivalent code in your preferred programming language. Please use any language other than Python

```
#include <string>
 using namespace std;

 int main() {

    enum names { John, Bob, Dave}
    String names2[3] = {'John', 'Bob', 'Dave'}

    for (int i = John; i <= Dave; i++)
        cout << i << names2[i];

 }
```