**CS-3310-Term-Project**

**Algorithm Pseudocode**

Decimal to Binary

func decToBin(int decNum)

{

string binNum = ""

while(decNum >0)

{

int remainder = decNum mod 2

binNum += to\_string(remainder)

decNum = decNum / 2

}

int j = 0

int i = 0

for (j = size of binNum - 1, i = 0; i <= j; j--, i++)

{

int tmp = binNum[j

binNum[j] = binNum[i]

binNum[i] = tmp

}

return binNum

}

Binary to Decimal

func binToDec(int binNum)

{

int decVal = 0

int baseVal = 1

int length = size of binNum

for (int i = length - 1; i >= 0; i--) {

if (binNum[i] == '1')

decVal = decVal + baseVal

baseVal = baseVal \* 2

}

return decVal

}

Decimal to Hexadecimal

func decToHex(int decNum)

{

string hex = “”

while decNum != 0

{

int remainder = 0

char ch

remainder = decNum mod 16

if remainder < 10

{

ch = remainder + 48

}

else

{

ch = remainder + 55

}

hex = hex + ch

decNum = decNum / 16

}

reverse hex

return hex

}

Hexadecimal to Decimal

func hexToDec(string hexVal)

{

int length = size of hexVal

int baseVal = 1

int decVal = 0

for(int i = length - 1;i >= 0;i = i - 1)

{

if hexVal[i] >= ‘0’ and hexVal[i] <= ‘9’

{

decVal = decVal + ((hexVal[i] - 48) \* baseVal)

baseVal = baseVal \* 16

}

else if hexVal[i] >= ‘A’ and hexVal[i] <= ‘F’

{

decVal = decVal + ((hexVal[i] - 55) \* baseVal)

baseVal = baseVal \* 16

}

}

return decVal

}

Hexadecimal to Binary

*(Utilizes call of above functions)*

func hexToBin(string hexNum)

{

int decEquivalent = hexToDec(hexNum)

string binNum = decToBin(decEquivalent)

return binNum

}

Binary to Hexadecimal

*(Utilizes call of above functions)*

func binToHex(string binNum)

{

int decEquivalent = binToDec(binNum)

string hexNum = decToHex(decEquivalent)

return hexNum

}

**Language Implementations**

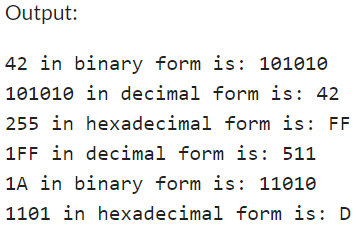
See attached files below for programming language implementations of pseudocode:

*Part2C++Code.cpp*

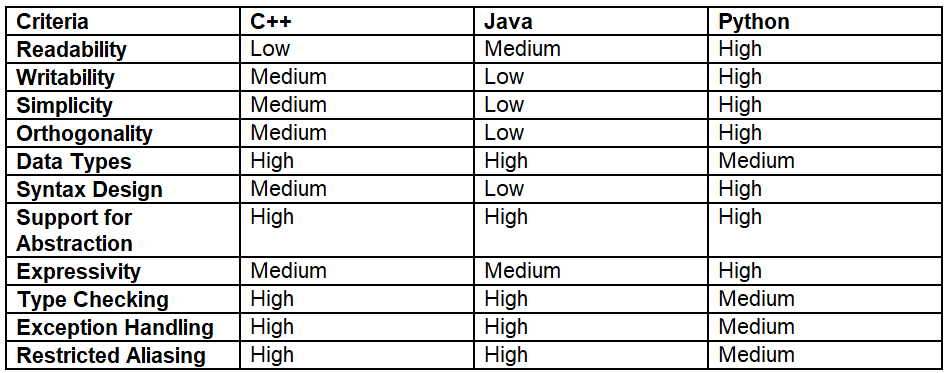
*Part2JavaCode.java*

*Part2PythonCode.py*

Output for the three programs:



**Language Evaluation Criteria**

Assessment: 

* Python shines for its high readability and writability, thanks to its English-like syntax and minimalistic design.
* C++ offers a balance with reasonable readability and efficient writability, alongside strong type checking and data type management.
* Java lags slightly due to its verbosity, which affects its readability and writability, despite strong type checking and exception handling.

**Reflection**

Ⅰ) Increased capacity to express ideas:

* C++: It's a language with vast library support and it allows for more control, especially in manipulating strings and characters. The use of libraries like <bits/stdc++.h> simplifies the implementation.
* Java: Java has a strong type system and class based approach that leads to more structured code. String manipulation is more verbose while also being clearer, as shown in the hexadecimal conversion method.
* Python: Python shines with its readability and simplicity. The easy to understand and concise syntax, as seen in list comprehensions and string operations, allows expressing complex ideas in fewer lines of code.

Ⅱ) Increased ability to learn new languages

* Concepts like data type conversion, string manipulation, and control structures are common but their implementation differs in each language. In programming with each language we obtained a deeper understanding of programming concepts beyond syntax.

Ⅲ) Better understanding of the significance of implementation

* Each language's unique features influence implementation. In Python, dynamic typing and builtin functions make certain operations more straightforward than in C++ or Java, where more code is required to achieve the same functionality. The choice of language can significantly impact the efficiency and complexity of the code. For example, reversing a string in Python can be done succinctly using slicing, whereas in Java and C++, it requires more lines of code and manual handling.

Ⅳ) Better use of languages that are already known

* C++: The experience with C++ in this project shows its strength in low-level operations and control over memory management, critical for performance intensive applications. The explicit handling of data types and manual string manipulation in C++ help us gain a deeper understanding of how operations are executed at a lower level.
* Java: Java's strong type system and object-oriented features encourage a more structured approach to problem solving. This experience helps in appreciating Java’s comprehensive standard libraries and its utility in developing robust and maintainable code for large-scale applications.
* Python: The experience with Python highlights its efficiency for rapid development and prototyping, which is especially useful for scripting and data analysis.

Ⅴ) Improved background for choosing appropriate languages

* This experience showed the importance of selecting the right tool for the job. For quick script-like tasks, Python's brevity is better. For applications requiring intensive memory and CPU usage, C++ might be the better choice due to its lower level capabilities. Java's portability and extensive libraries make it a strong candidate for cross platform applications and large scale systems.

**Responsibility Table**

| *Team Member Name* | *% Contribution* |
| --- | --- |
| Arturo Perianez |  |
| Brynne Wright |  |
| Hunter Lane |  |