Assignment 1

1. Briefly explain the main criteria in evaluating a programming language. Compare two programming languages of your choice based on these criteria.

Ans: There are several main criteria in evaluating a programming language, and each criteria is judged based on three functionality;

* Readability: Readability is used to assess how easy it is to understand code written in a programming language just by reading or observing it. This includes non-programmers, they should at least understand the controls and workflow.
* Writability: Writability refers to how easy it is to write a program using a language. As it is with normal languages, program languages face a similar problem of complex syntaxes. The better the writability, the better the readability.
* Reliability: A program is said to be reliable if it is working according to its functionality in all circumstances. It assures that the program is not behaving unexpectedly.

The criteria includes:

1. Simplicity: If a programming language is simple, it increases the readability and writability, which enhances reliability.
2. Orthogonality: Using different features to build a combination which then gives a meaning is known as Orthogonality.
3. Data type and Structure: The concept of any programming language is built upon data types and structures. Data types are categorized where different types of field elements are grouped together.
4. Control Structure and Syntax Design: They have a specific format which are also basics of languages. Defining a set of rules like grammar in english language that's what Syntax Design is.
5. Type Checking: Conversion of Data types defines Type Checking. Although it is a bit complex for new users, all the language tends to grab users attention in type checking.
6. Exception handling: Handling unusual situations while the program is in runtime falls into the hands of Exception Handling capability of a program.
7. Aliasing: Concept of referencing is used in aliasing, i.e. giving two variables the same memory addresses. That makes it difficult to perform certain operations on either of those variables.
8. Abstraction: It refers to privacy of important code or just organizing the code in a very efficient manner. Although it helps programmers to implement the functionality, for programming languages it is a complicated procedure.

Generalizing on all of the criterias, the two programming languages I’ve chosen to compare are C and C++.

There is no comparison when you take C language against anything but its successor C++. Over the years C has evolved to C++ but there still exists some tasks which can only be done by C. C language has low readability and writability compared to C++ but it has more reliability. There is no language complicated then C, except assembly and machine language. That's how it gets to be the fastest performing language. C++ is more simpler and can easily be made complex using its features. Exception handling can be easily definable using C++ rather C. Syntax of C++ is simpler than C. Aliasing is one of the hardest concepts in C and that's the reason most programmers prefer different languages. One of the key points in upgrading to C++ from C is Anti Aliasing.

2. Discuss three possible ways to classify programming languages into different classes. Explain what are the main functions of each of the classification methods.

Ans: There are two main categories where different classification of programming languages are seperated:

1. Low level language: This category includes Machine language and Assembly language
2. High level language: This category includes Procedural-Oriented language, Problem-Oriented language and Natural language.

* **Machine Language**: This is the language which runs and works with hardware directly. Almost all programming languages convert itself to machine language before running as this is the only way programs get what it needs from the computer. Machine language is a set of 0’s and 1’s in a specific combination which defines the circuit instructions as current or no current (P.S. This is not how machine language behaves but to understand the concept it is generalized).
* **Assembly Language**: The language 1 step more readable than ML is Assembly language. It generalizes combinations of 1’s and 0’s which define certain operations like Addition, print and ending certain loops to short hands ‘ADD’, ‘PRINT’ and ‘END’ respectively. The assembler then converts these short hands to machine language to run the program.
* **Problem-Oriented Language**: This language allows users to write short code to perform more stuff than other predecessor languages. The main use case is the easiness of giving the result which the user(programmer) asked for without letting them configure how the result is cooked. This is why beginner finds VB, C#, PHP is much easier than C, FORTRAN, etc. Also the syntax is close to human language so the instructions given are more easier to decode.

3. Describe three perspectives/views on programming languages (i.e. user, designer, and implementer).

Ans:

* User: User is the one all of the products are made. Even considering the programming language as a product, there are programmers as users which decide which is better. When we talk about users in general, in most cases they don’t care about how their product was made but are focused and care just to use the product. As a user it is better to know at least on what programming language your product was built.
* Designer: Although they play a big role in the software life cycle, they don’t necessarily need to have any programming language. In the ongoing world you can build pretty much anything you want just if you have design and concept ready. There are lot of softwares available which allows you to do that. Although that is not enough for a product to be sellable. That’s where a product needs a professional designer as it gives a face to the functionality. There are two types of designers/developers, front-end and back-end. Both of these designer’s are required to compose a product. Generally front-end developers are known as designer’s and back-end designer’s as developers.
* Implementer: They are the heart of a product. A product's needs and features and use cases are built by Implementers. They are the logical coders which deal with the core concept of the product and are the reason the product is living, all others are just variables in the cloud. Generally they are the backend developers which deals with programming languages like php, C, C#, JavaScript(Partially) and also works with databases. They also implement the security to the code and the product. They are responsible for the majority of testing and updates.

4. Find an article or a book chapter on the philosophy or principles of UNIX operating systems. Discuss two particular principles and explain their connections to programming languages.

Ans: The book I chose was ‘Linux with Operating System Concepts By Richard Fox’. It was not a big read but has a comprehensive explanation of Linux OP. The books answered so many questions and is a good source for beginners who have operating system background. The two principles of linux discussed were Rule of Morality and Rule of Charity. Rule of Morality approaches writing code in parts which should be connected by clean intercepts. Rule of charity says that Charity is better than Cleanliness. As we see in C++ that the interface is the signature of multiple properties of methods, events or indexes. Also a class is the good example of a property implementing that interface have to have members of the class implemented which are included in the interface definition. Class must implement a method named constructor or sample method which takes no parameters and returns null or itself. As we talk about charity, functional concepts came into C++ through live and lambda expression. There are a lot of other functionalities in C++ we can discuss when it comes to the principals of the UNIX operating system.

6. Read the paper: Donald E. Knuth, Literate Programming. Give a brief summary of "Literate Programming."

Ans: The paper was published in 1983 which makes it so impressive as someone can come up with such an amazing idea with no real implication seen before. The paper was written in a funny but convincing manner which makes it enjoyable as well. The content was about WEB as a generalized local form. As he mentioned the WEB is built up on an ideology of building a complex structure or set of rules and other objects which correlates and has information of its neighbors. The WEB initially was incorporated using two different languages. Donald said that by using two different languages to comprehend a program does not undermine the quality or functionality of a language but enhances it by using separate languages to integrate together, the limitations made irrelevant. The two languages he chose were TEX as a document formatting language and PASCAL as the programming language. Comparing both of these languages to current day, TEX is close to HTML and PASCAL is close to JavaScript. He chose TEX because he was the one who invented it and PASCAL was his choice of Programming Language because at the time it was everyone’s second language to learn. Also WEB offers Micro services which makes PASCAL’s limitation irrelevant.

In the paper he also describes the approach of how WEB works and behind the scenes how it compiles the code. The two approaches he described were WEAVE and TANGLE. WEAVE is a process called weaving which takes a WEB document and converts it to TEX file. Further TEX processor converts the file to DVI output file in binary format which describes the typeset of documentation.The tangling process is similar to weaving but instead it takes WEB document and converts it into the PAS files which then uses PASCAL compiler to compile the file to REL. The executable REL file can be used to perform the instructions.

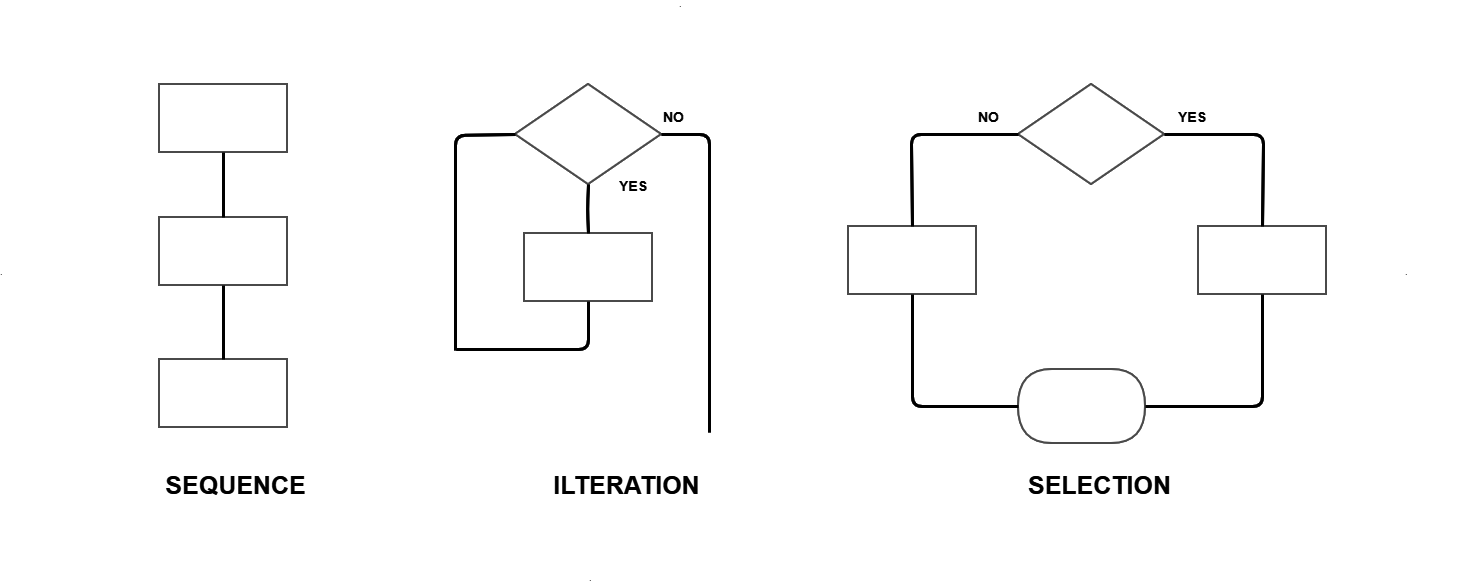
Further the paper describes more detailed information about the process. In conclusion, this was the paper which changed the world we live in. Although the paper was old, it still holds the value that it had back in 1983.

7. Describe influences of computer architectures and programming methodology on the development of programming languages.

Ans: In this day and age, there are a lot of different computer architectures and their different implementations over different OS. But oddly they don’t influence the development of programming languages as much. Computer architecture might influence complexity for programming language in terms of efficiency. This is because most programming languages are designed as interpreted languages which use compilers to compile their code. And almost all the interpreted languages use C or C++ for compilation.

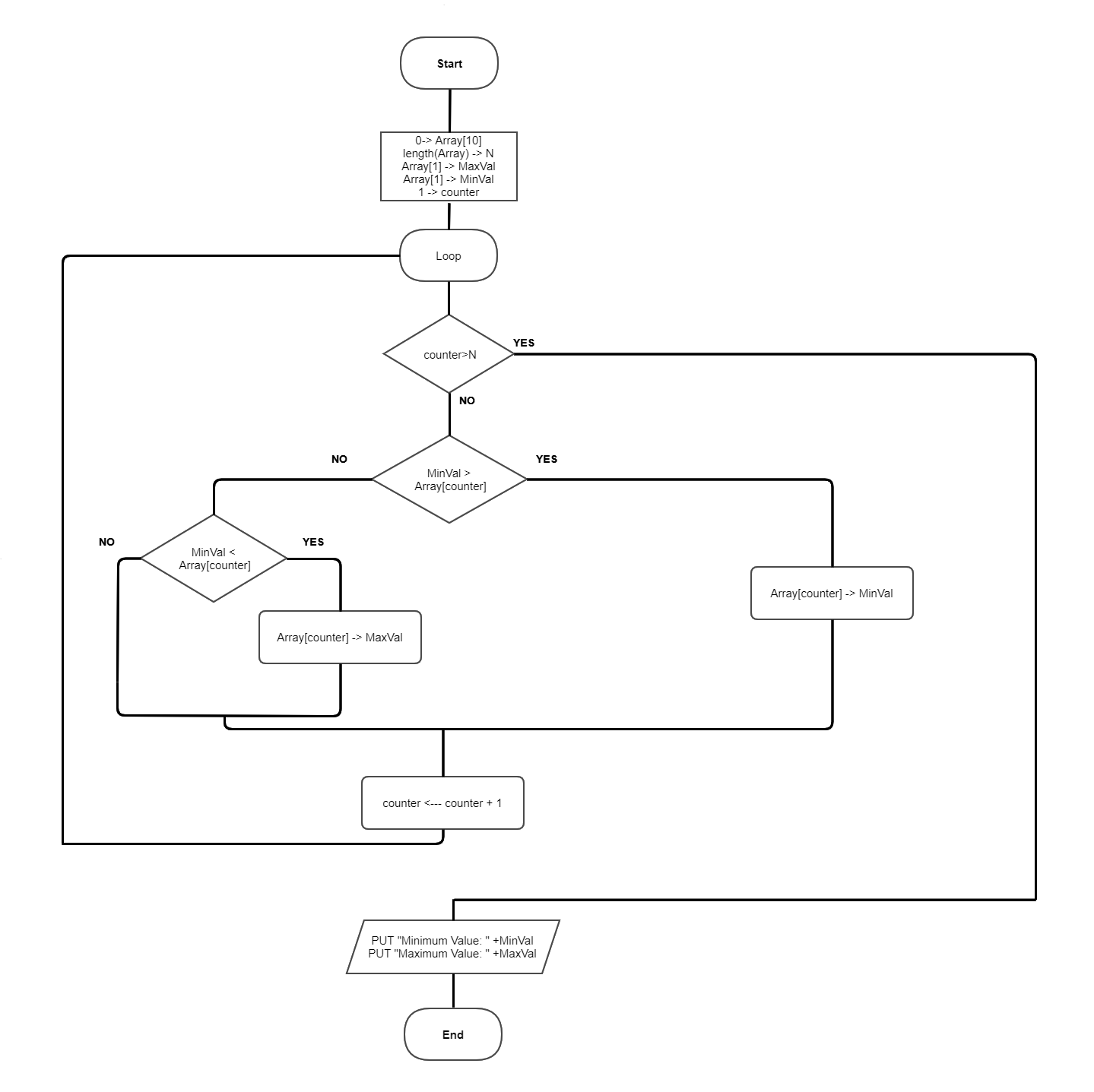
When we talk about programming methodology, one of the biggest changes in the computer world was the introduction to Object Oriented Programming. This is where the new era begins as the use case is so drastics that all the new and old programming languages started to implement OOP. This has influenced others to discover something so drastic and game changer like OOP. This helps us as users to see more broad ideas and all kinds of different implications of programming like Functional Programming and Procedural Programming.

8. Draw float charts of three basic control structures.

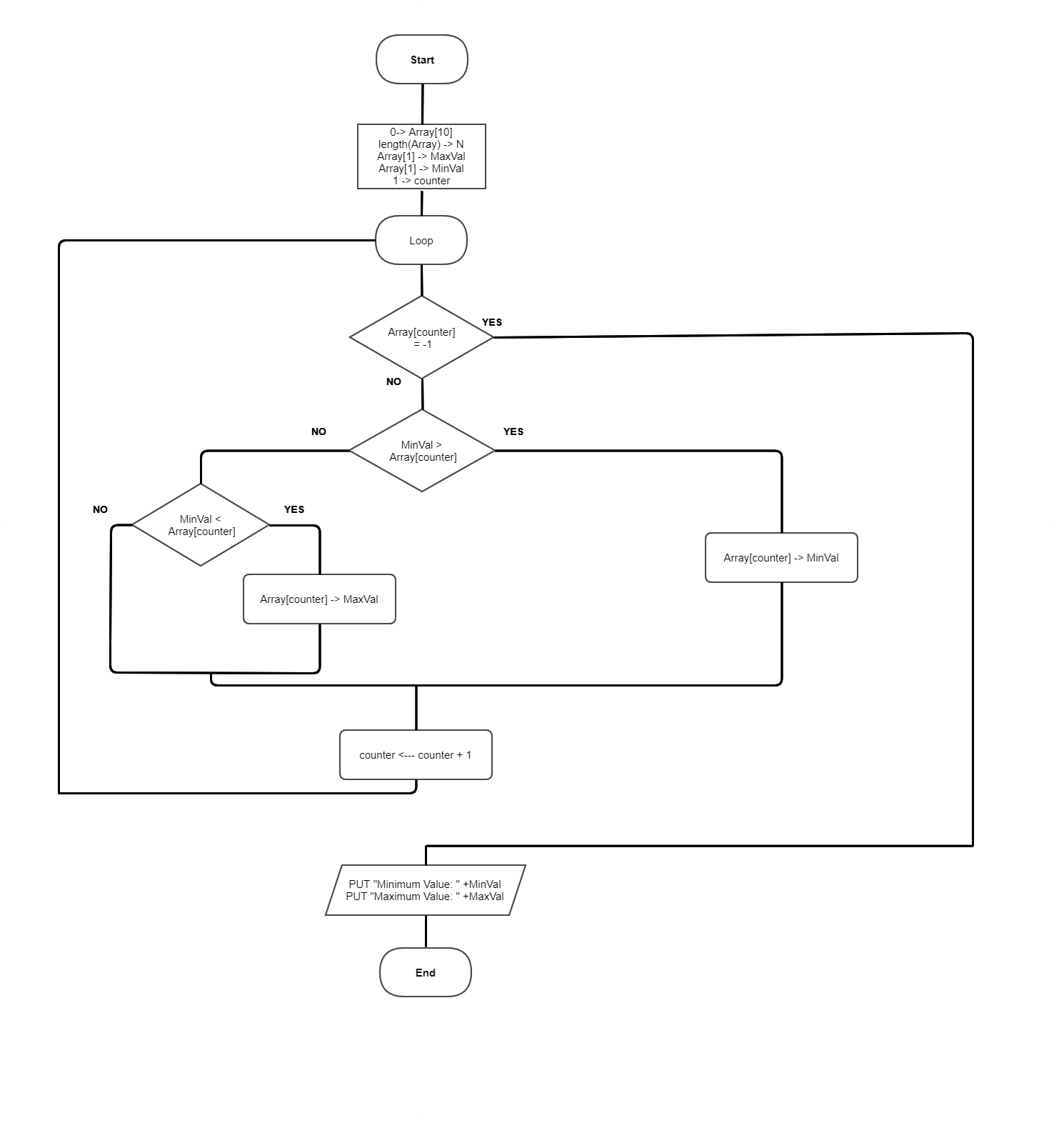


9. Draw a flowchart to find the maximum value and the minimum value of a list of N numbers, where N >= 1. Identify the basic control blocks in your flowchart.

Ans:



10. Draw a flowchart to find the maximum value and the minimum value of a list of positive numbers, where -1 indicates the end of the list. Identify the basic control blocks in your flowchart.

Ans:

Cited Work:

* Linux with Operating System Concepts By Richard Fox
* Literate Programming By Donald E. Knuth
* Cacoo.com (making flow chart)