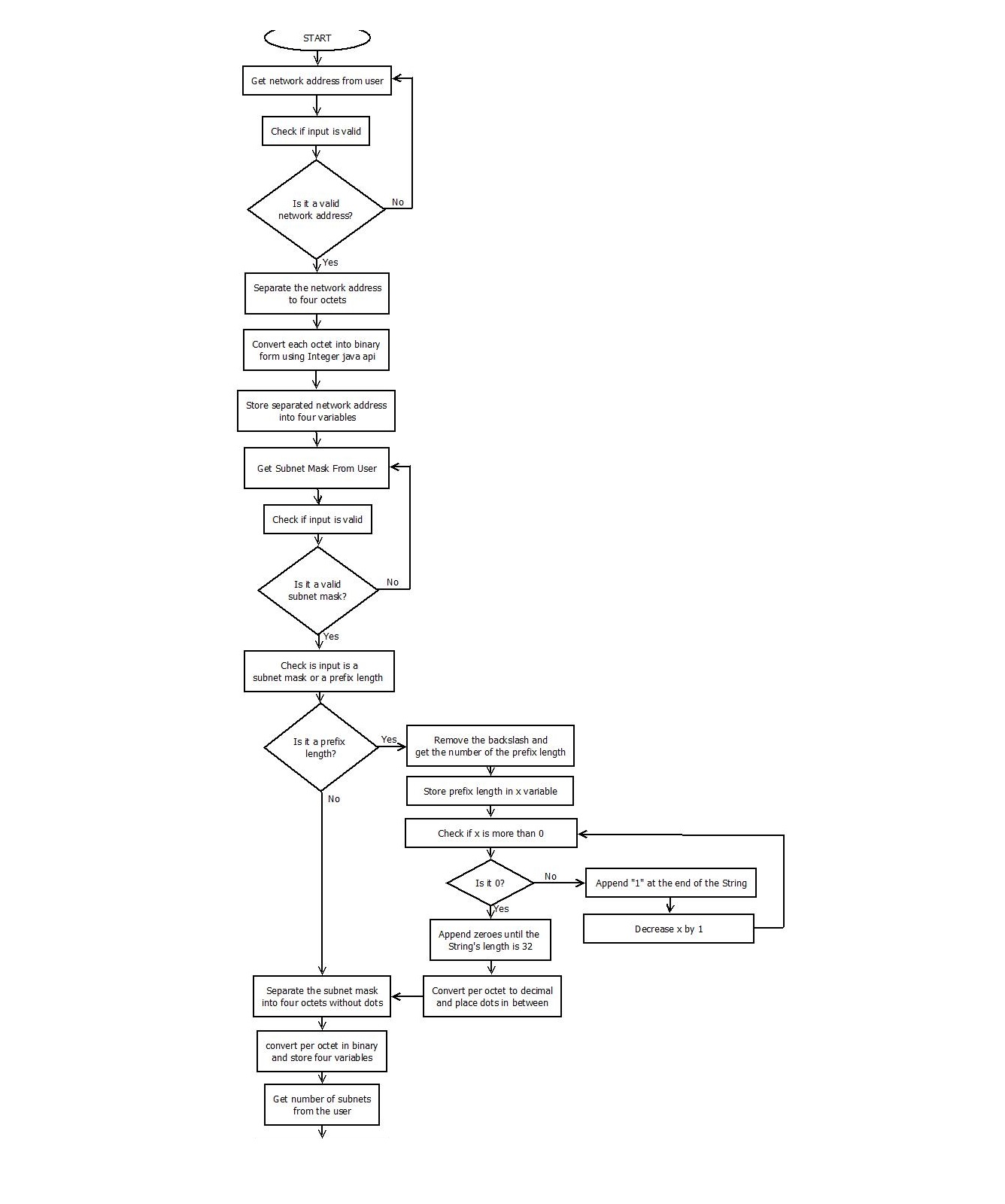
INTR-NW

Machine Project Documentation

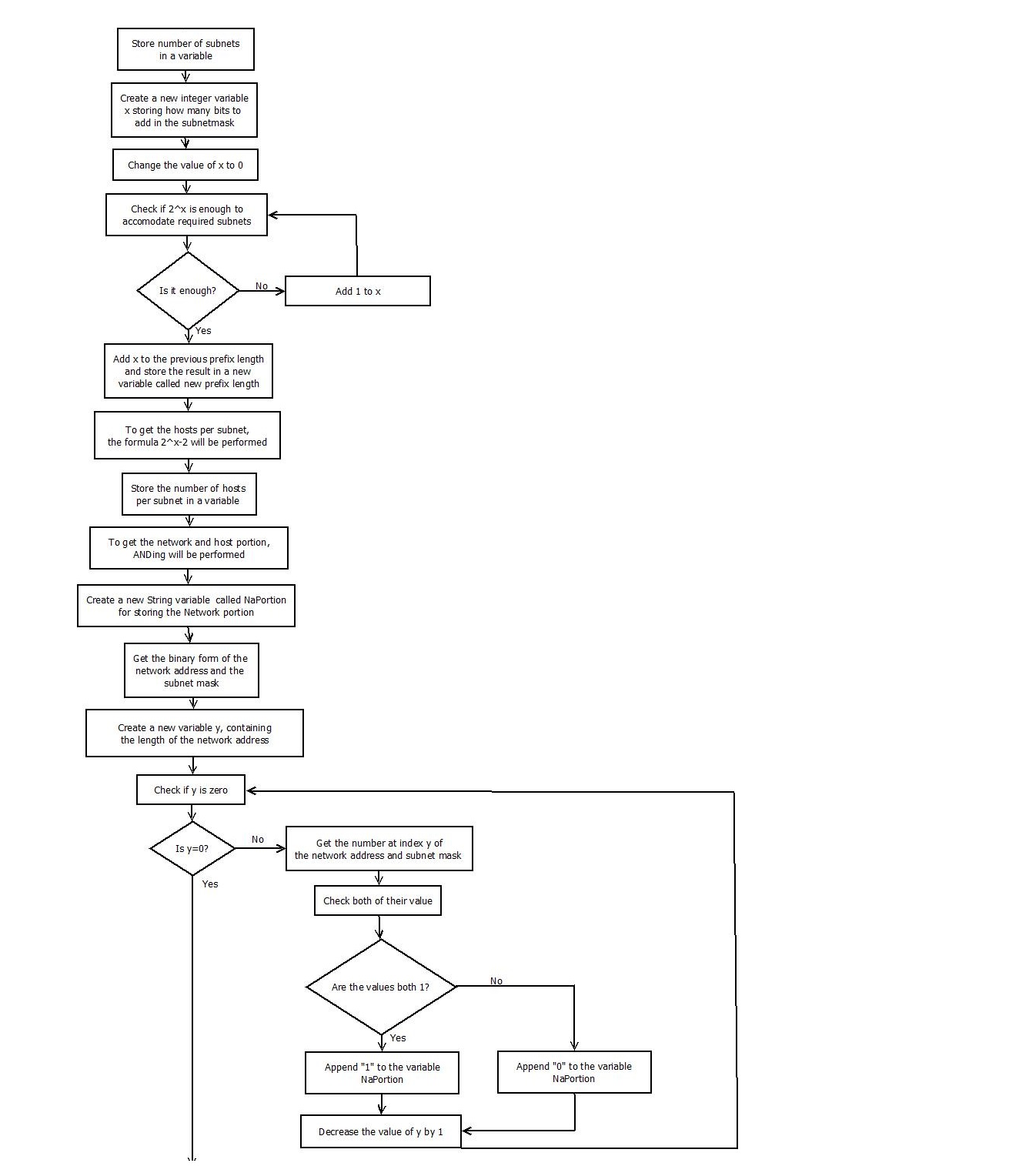
SUBMITTED BY:

**Lim, Ivana Koon Yee U.**

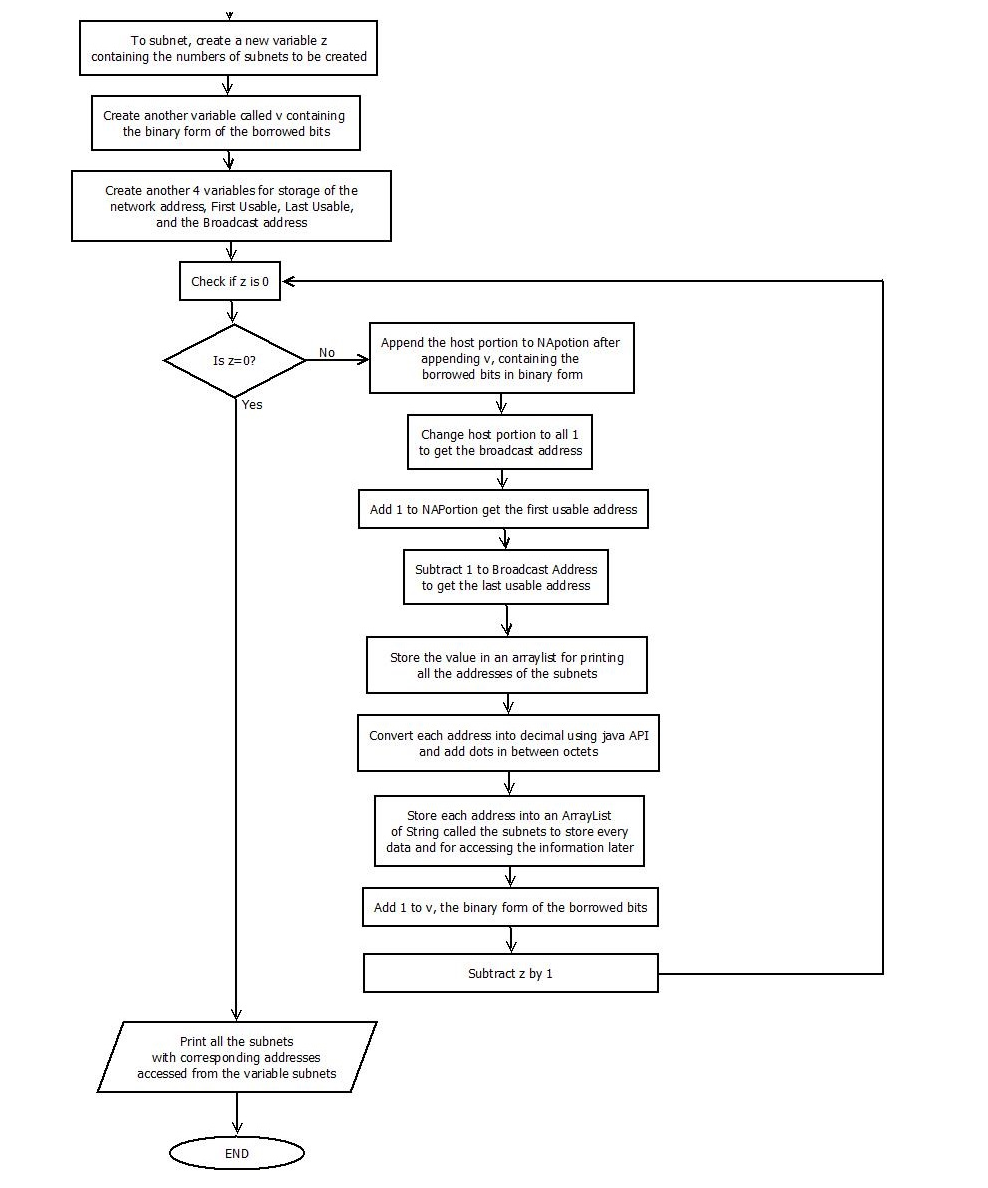
**Tan, Carl Aaron**

System Design

**Figure 1. Subnetting Calculator Partition 1**

1. System Design
2. System Desig

**Figure 2. Subnetting Calculator Partition 2**



**Figure 3. Subnetting Calculator Partition 3**

Discussion

The subnet calculator works by getting the original network address (ONA), original subnet mask (OSM), and the number of subnets from the user. The program will then convert the decimal values of the ONA and OSM into binary values and store them to their respective variables. The subnetting proper executes by getting the number of bits to borrow with respect to the entered number of subnets needed, and separating the network address portion (NAP) string from the host address portion (HAP) string. The NAP is obtained by a “pairing check” loop with the converted binary subnet mask, and the converted binary network address (similar to the actual process of ANDing). The HAP, on the other hand, is gained by subtracting the bits to be borrowed string number from the original HAP string. The assigning of subnet addresses is done inside a loop, appending the incrementing bits borrowed in between the NAP and the HAP, together producing the new network address of the subnet. Correspondingly, the broadcast address (BA), first usable address (FA), and last usable address (LA) are obtained by isolating the host address portion string inside the subnetting loop. The BA is gained by transforming the HAP string to an all-1s string for the BA, and for the FA and LA, the last bit of the HAP is truncated and replaced with a 1 for FA and the last bit of the BA is truncated and replaced with a 0 for the LA respectively.

1. Classes

|  |
| --- |
| **Class Subnetting** |
| **public Subnetting(String origNA, String subnetMask, int numOfSubnets)**  This is the constructor of the Subnetting class. It is required to build the object. |
| Parameters:  **origNA** - network address derived from the input of the user  **subnetMask** - subnet mask from the input of the user  **numOfSubnets** - number of subnets required by the user |
| Returns:  **N/A. This is the constructor of the class.** |

|  |
| --- |
| **public boolean checkNetworkAddress(String networkAddress)**  This method checks whether the network address is valid |
| Parameters:  **networkAddress** - String containing the network address |
| Returns:  **Boolean. true if network address is valid, otherwise returns false** |

|  |
| --- |
| **public int numOf1(String n)**  This method counts how many times the number 1 appears in a String |
| Parameters:  **n** - String value |
| Returns:  **Integer value containing how many times the number 1 occurred** |

|  |
| --- |
| **public java.lang.String getNewNetMask()**  This returns the value of the new Netmask |
| Parameters:  **n/a** |
| Returns:  **String of the new Netmask** |

|  |
| --- |
| **public String getNewHostsPerNet()**  This returns the value of the Hosts per subnet |
| Parameters:  **n/a** |
| Returns:  **String representing the number of hosts per subnet** |

|  |
| --- |
| **public java.util.ArrayList<String> getSubnets()**  This method computes the subnet and returns an arraylist of strings containing the Newtwork Address, First Usable Address, Last Usable Address, and the Broadcast Address of each subnet |
| Parameters:  **n/a** |
| Returns:  **an Arraylist of String containing all the Subnets** |

|  |
| --- |
| **public int binaryToDecimal(int binaryN)**  This method converts a binary value to a decimal number |
| Parameters:  **binaryN** - the binary value of number being converted |
| Returns:  **an Integer containing the decimal value** |

|  |
| --- |
| **public String decimalToBinary(int decimalN)**  This method converts a decimal number to a binary value |
| Parameters:  **decimalN** - the decimal number of the binary being converted |
| Returns:  **a String containing the binary value** |

|  |
| --- |
| **public String binaryFormWithPeriod(int n1, int n2, int n3, int n4)**  This combines four octets and places a dot after each octet |
| Parameters:  **n1** - the first octet  **n2** - the second octet  **n3** - the third octet  **n4** - the fourth octet |
| Returns:  **String containing the merged octets with dot as separator** |

|  |
| --- |
| **public String binaryFormNoPeriod(int n1, int n2, int n3, int n4)**  This combines four octets into a Single String value |
| Parameters:  **n1** - the first octet  **n2** - the second octet  **n3** - the third octet  **n4** - the fourth octet |
| Returns:  **String containing the merged octets without dots between each octet** |

|  |
| --- |
| **public int netmask(int n)**  This method computes the netmask |
| Parameters:  **n** - a binary value |
| Returns:  **Integer value containing the new Netmask** |

|  |
| --- |
| **public int bitsToAdd(int numOfSubnets)**  This method computes how many bits are needed to be borrowed |
| Parameters:  **numOfSubnets** - the number of subnets specified by the user |
| Returns:  **Integer value containing how many bits are borrowed** |

|  |
| --- |
| **public int hostsPerNet(int numOfSubnets)**  This computes how many hosts per net are available per subnet |
| Parameters:  **numOfSubnets** - the number of subnets specified by the user |
| Returns:  **Integer value containing the number of hosts per subnet** |

|  |
| --- |
| **public String numOf1ToDecimal(int n)**  This method converts an integer value into a binary containing several 1's and converting it back to a decimal value with dots separating each octet |
| Parameters:  **n** - the Integer to be converted |
| Returns:  **String value containing the decimal value** |

|  |
| --- |
| **public String numOf1ToBinary(int n)**  This method converts an integer value into a binary containing several 1's with periods |
| Parameters:  **n** - String value |
| Returns:  **Integer value containing how many times the number 1 occurred** |

|  |
| --- |
| **public String IPadd(java.lang.String n)**  This method converts an binary String into a decimal into an IP address format |
| Parameters:  **n** - the binary String to be converted |
| Returns:  **the converted binary String into an IP address format** |