CPSC1150 – Lab 5

by Cordell Bonnieux - 100372192

Binary to Decimal

Program: B2D.java

Flowchart: B2D.jpg

This program reads a string from the user, determines if it is a valid binary number. If so, converts that binary number to decimal format and prints to console. If not, the program restarts.

Pseudocode:

START

PRINT “Enter a binary number”

READ *binary*

COMPUTE integers: *decimal* = 0,

FOR integer *i* = length of *binary* and *e* = 1 **AS LONG AS** *i* > 0 **THEN** *i*-- and *e* *= 2

    IF *binary* index *i* **EQUALS** 1

        decimal += *e*

    ELSE IF *binary* index *i* **DOES NOT EQUAL** 0

        PRINT *binary* “ is not a valid binary number”

        RESTART PROGRAM

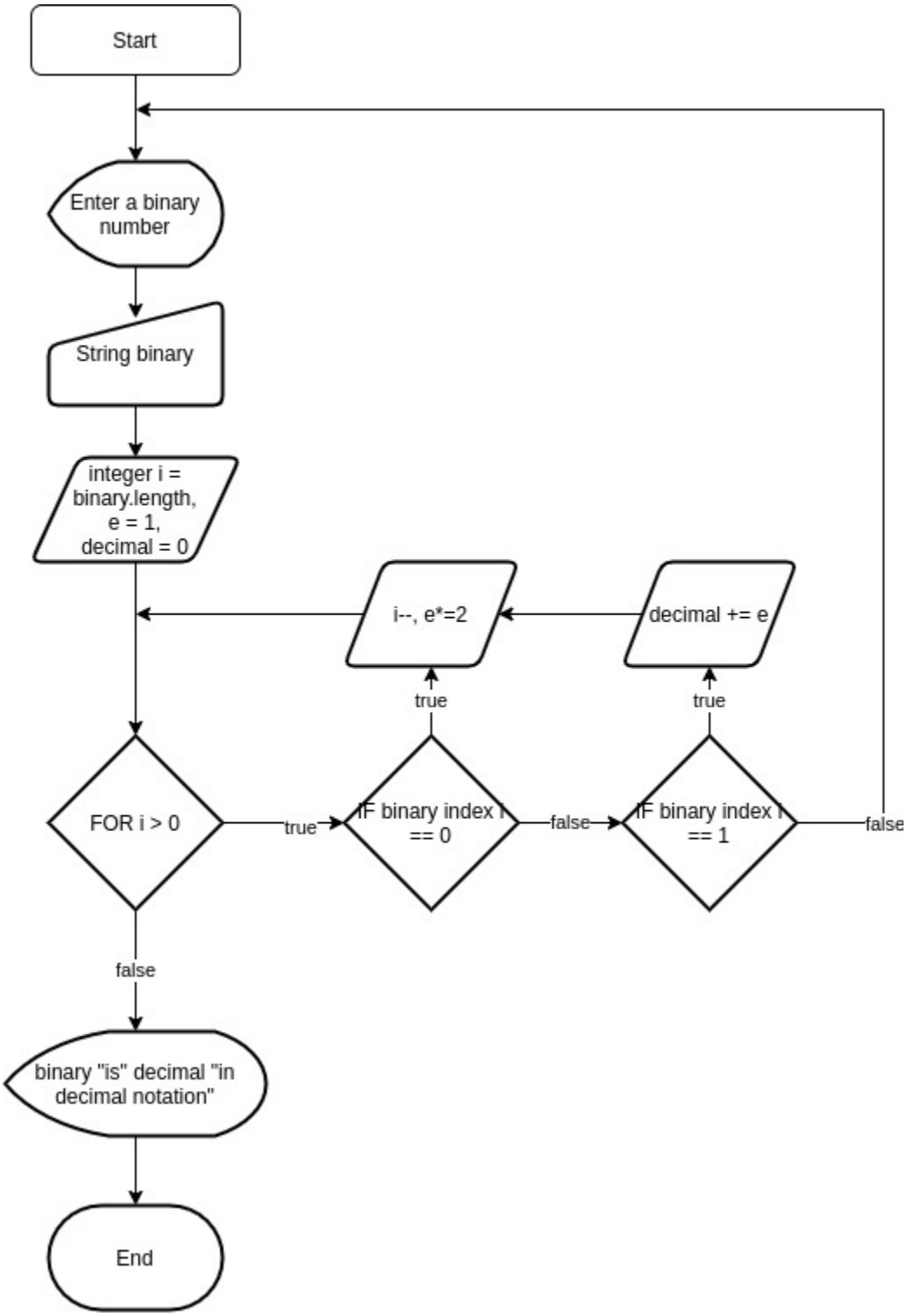
PRINT “the binary number” *binary* “ in decimal notation is” *decimal*

END

Test Case:

```
-11-openjdk-amd64/bin/java -Dfile.encoding=UTF-8 -cp "/home/
Enter a binary number: 1117786
1117786 is not a valid binary number.
Enter a binary number: fake number 112
fake number 112 is not a valid binary number.
Enter a binary number: not a number?
not a number? is not a valid binary number.
Enter a binary number: 11100011121111
11100011121111 is not a valid binary number.
Enter a binary number: 10101110011
the binary number 10101110011 in decimal notation is 1395.
```

Flowchart:


Start
Enter a binary number
String binary
integer i = binary.length, e = 1, decimal = 0
FOR i > 0
true
IF binary index i == 0
false
IF binary index i == 1
false
true
decimal += e
true
i--, e*=2
false
binary "is" decimal "in decimal notation"
End

Primes

Program: Primes.java

This program displays a table in the console showing the prime numbers between 2 and 1000 inclusive.

Pseudocode:

START

COMPUTE Array *primes*

FOR integer *i* = 1000 **AS LONG AS** *i* > 1 **THEN** *i*--

    COMPUTE boolean *isComposite* = false

    FOR integer *y* = 2 **AS LONG AS** y <= *i* **THEN** *y*++

        IF *i* % *y* == 0

            COMPUTE *isComposite* = true

    IF *isComposite* **IS FALSE**

        COMPUTE *primes* += *i (push i to primes)*

PRINT “Prime numbers from 2- 1000:”

COMPUTE integers *primeCount* = *primes*.length(or size), *x* = 0

WHILE *x* < *primeCount*

    PRINT formatted: primes.x++, primes.x++, primes.x++, primes.x++, primes.x++, primes.x++, primes.x++, primes.x++

Test Case:

```
cordell@cordell-ROG-Zephyrus-G14-GA401IU-GA401IU:~/Do
-11-openjdk-amd64/bin/java -Dfile.encoding=UTF-8 -cp
Prime numbers from 2 - 1000 (inclusive):
997 991 983 977 971 967 953 947
941 937 929 919 911 907 887 883
881 877 863 859 857 853 839 829
827 823 821 811 809 797 787 773
769 761 757 751 743 739 733 727
719 709 701 691 683 677 673 661
659 653 647 643 641 631 619 617
613 607 601 599 593 587 577 571
569 563 557 547 541 523 521 509
503 499 491 487 479 467 463 461
457 449 443 439 433 431 421 419
409 401 397 389 383 379 373 367
359 353 349 347 337 331 317 313
311 307 293 283 281 277 271 269
263 257 251 241 239 233 229 227
223 211 199 197 193 191 181 179
173 167 163 157 151 149 139 137
131 127 113 109 107 103 101 97
89 83 79 73 71 67 61 59
53 47 43 41 37 31 29 23
19 17 13 11 7 5 3 2
```