Alexandria University
Faculty of Engineering
Computer and Systems Engineering
Department
CSE 214: Discrete Structures

Lab 3 – Number Theory

Team Members

1. ID: 20011574 – محمد رفيق محمد السيد ابراهيم
2. ID: 20011923 – مصطفى خالد كمال احمد زايد

Question One:

First: Problem Statement

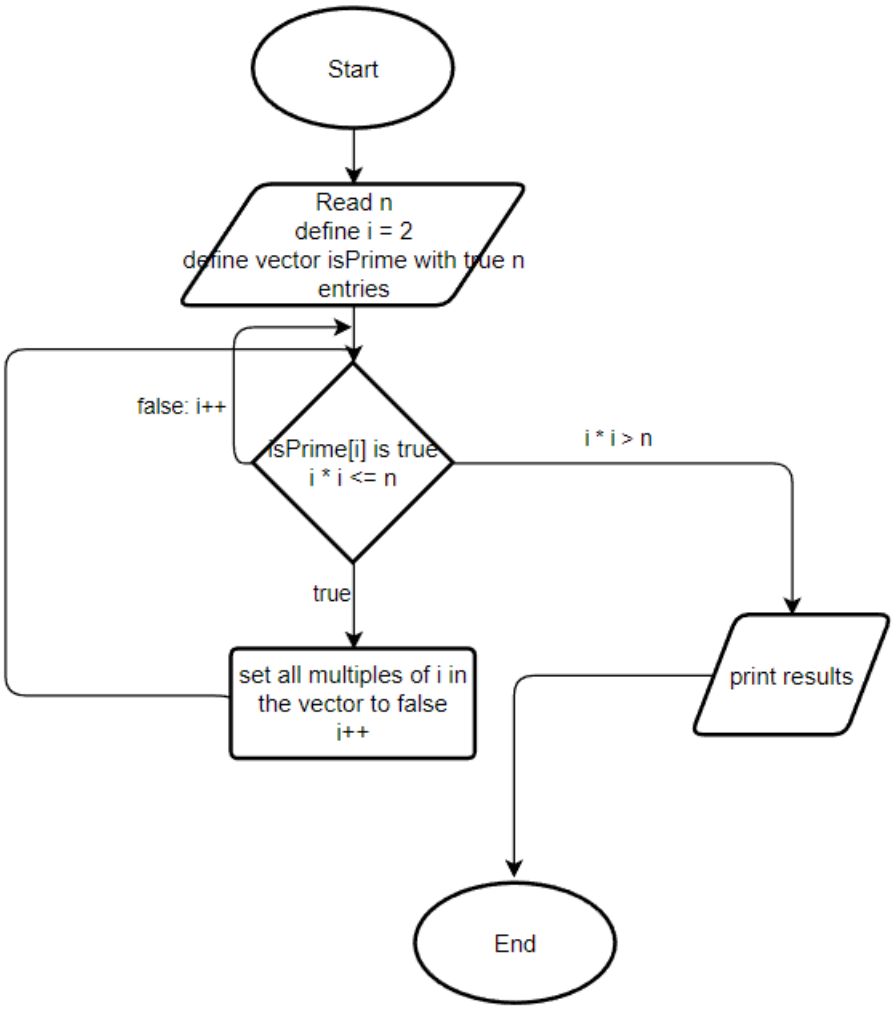
It is required to implement sieve of Eratosthenes algorithm for finding all prime numbers up to any given limit.

Second: Used Data Structures

Vector:

isPrime<>: used to store the prime numbers less that or equal to the give input.

Third: Flow Charts



Fourth: Assumptions

We assumed the input will not exceed the C plus plus int bound.

Fifth: Design Decisions

A very straight forward implementation of the sieve algorithm, the only decision we made was using a vector to store the results since it is easy to deal with and easy to preset.

Fifth: Sample Runs

```
Enter the number to find primes less that or equal to it: 100
Prime numbers less than or equal to 100 are:
2 3 5 7 11 13 17 19 23 29
31 37 41 43 47 53 59 61 67 71
73 79 83 89 97
Process returned 0 (0x0)   execution time : 4.187 s
Press any key to continue.
```

```
Enter the number to find primes less that or equal to it: 1000
Prime numbers less than or equal to 1000 are:
2 3 5 7 11 13 17 19 23 29
31 37 41 43 47 53 59 61 67 71
73 79 83 89 97 101 103 107 109 113
127 131 137 139 149 151 157 163 167 173
179 181 191 193 197 199 211 223 227 229
233 239 241 251 257 263 269 271 277 281
283 293 307 311 313 317 331 337 347 349
353 359 367 373 379 383 389 397 401 409
419 421 431 433 439 443 449 457 461 463
467 479 487 491 499 503 509 521 523 541
547 557 563 569 571 577 587 593 599 601
607 613 617 619 631 641 643 647 653 659
661 673 677 683 691 701 709 719 727 733
739 743 751 757 761 769 773 787 797 809
811 821 823 827 829 839 853 857 859 863
877 881 883 887 907 911 919 929 937 941
947 953 967 971 977 983 991 997
Process returned 0 (0x0)   execution time : 2.805 s
Press any key to continue.
```

Question Two:

First: Problem Statement

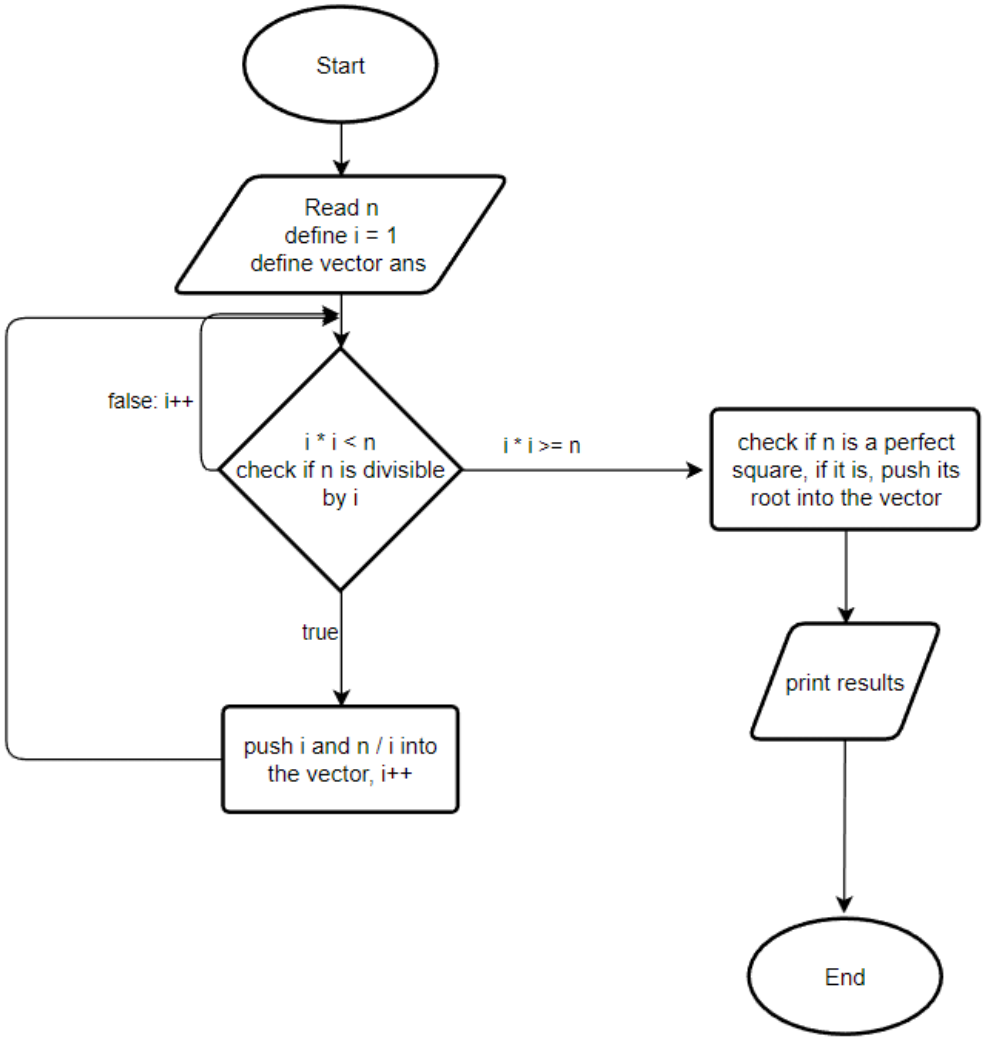
It is required to implement a trial division algorithm to find all possible integer factors to a given number.

Second: Used Data Structures

Vector:

ans<>: is used to store the resulting factors of the given number.

Third: Flow Charts



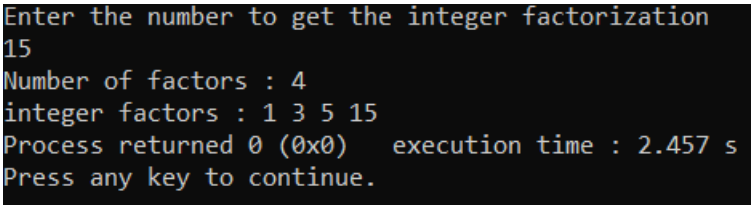
Fourth: Assumptions

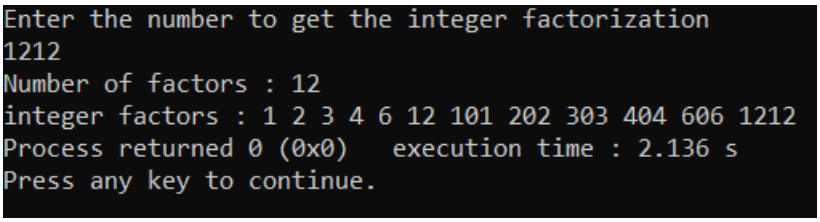
We assumed the input will not exceed the C plus plus int bound.

Fifth: Design Decisions

A very straight forward implementation of the trial division algorithm, and similar to the first question, the only decision we made was using a vector to store the results since it is easy to deal with and easy to preset.

Fifth: Sample Runs





Question Three:

First: Problem Statement

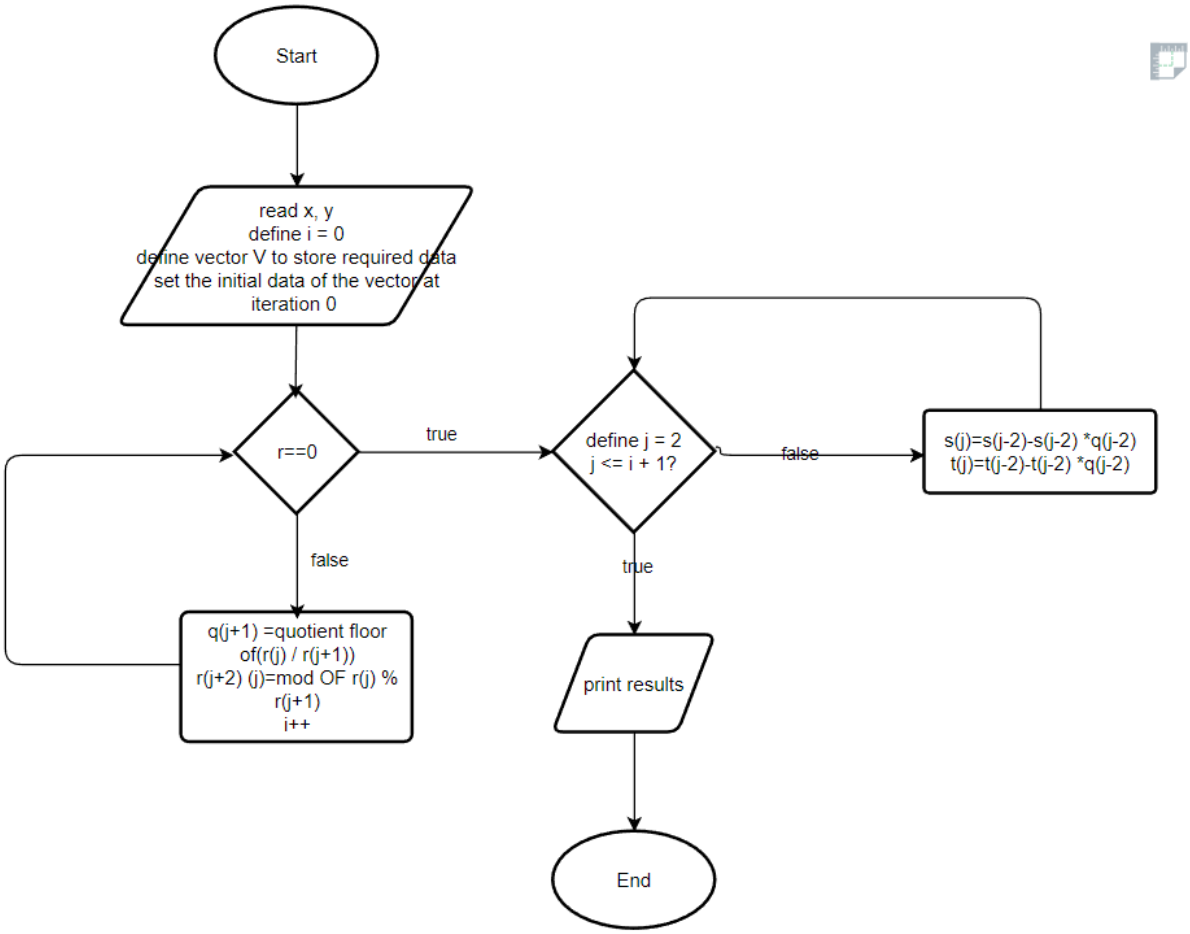
It is required to implement the extended Euclidean algorithm that finds the greatest common divisor d of two positive integers a and b. In addition, it outputs Bezout's coefficients s and t such that d = s a + t b

Second: Used Data Structures

Vector:

V<>: a vector of vectors used to store the required values of each iteration during the evaluation of the Euclidean algorithm.
The details of this vector are clearly shown in the code.

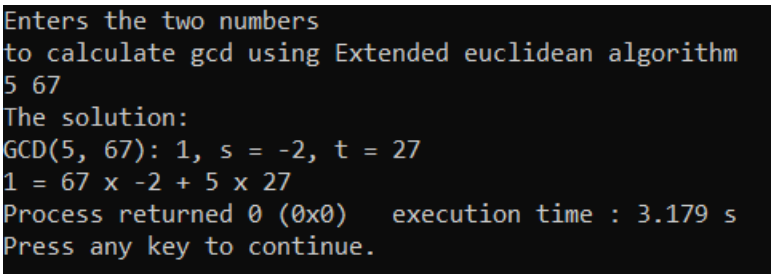
Third: Flow Charts

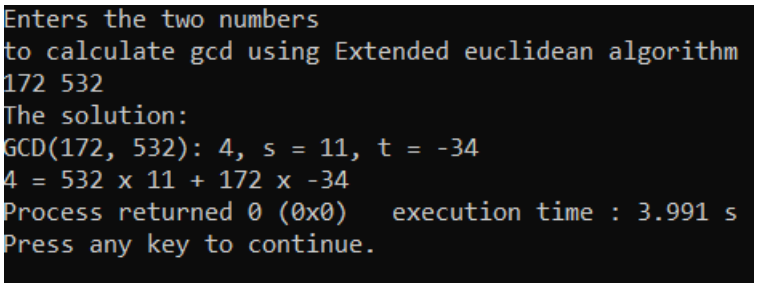


Fourth: Assumptions

We assumed the input will not exceed the C plus plus int bound.

Fifth: Sample Runs





Question Four:

First: Problem Statement

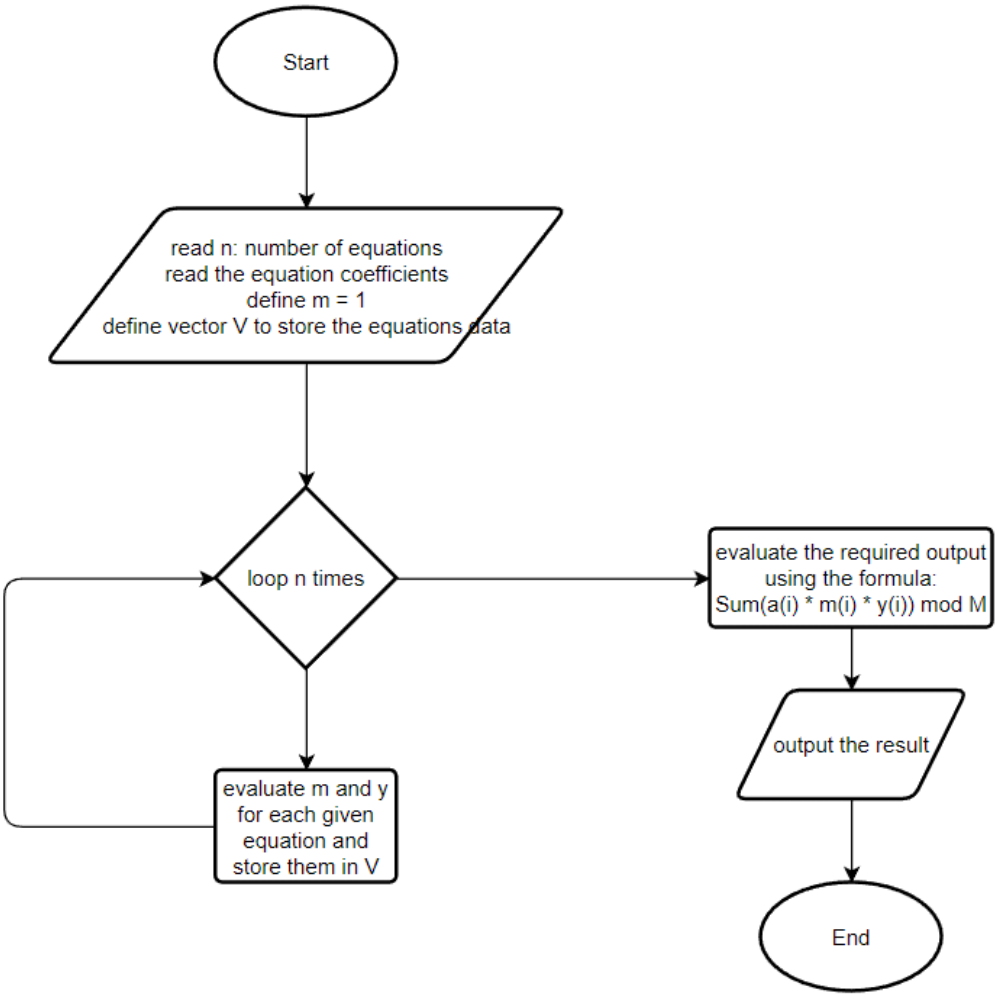
It is required to implement Chinese remainder theorem that takes as input m1, m2, m3, ..., mn that are pairwise relatively prime and (a1, a2, ..., an) and calculates x such that:
x = a1 (mod m1)
x = a2 (mod m2)
...
x= an (mod mn)

Second: Used Data Structures

Vector:

V<>: used to store the parameters and coefficients of each given equation.

Third: Flow Charts



Fourth: Assumptions

We assumed the user will follow the restrictions specified in the problem statement.

Fifth: Design Decisions

We decided to use the extended Euclidean theorem that we implemented in the previous question to find the inverse y of m for every iteration.

Fifth: Sample Runs

```
Enter number of equations: 3
Enter the value of the equivalence(a0): 46
Enter the modulo of the equivalence(m0): 2
Enter the value of the equivalence(a1): 63
Enter the modulo of the equivalence(m1): 5
Enter the value of the equivalence(a2): 66
Enter the modulo of the equivalence(m2): 7
The solution is : 38

Process returned 0 (0x0)   execution time : 23.210 s
Press any key to continue.
```

```
Enter number of equations: 4
Enter the value of the equivalence(a0): 5
Enter the modulo of the equivalence(m0): 2
Enter the value of the equivalence(a1): 7
Enter the modulo of the equivalence(m1): 3
Enter the value of the equivalence(a2): 45
Enter the modulo of the equivalence(m2): 5
Enter the value of the equivalence(a3): 66
Enter the modulo of the equivalence(m3): 7
The solution is : 115

Process returned 0 (0x0)   execution time : 16.999 s
Press any key to continue.
```

Question Five:

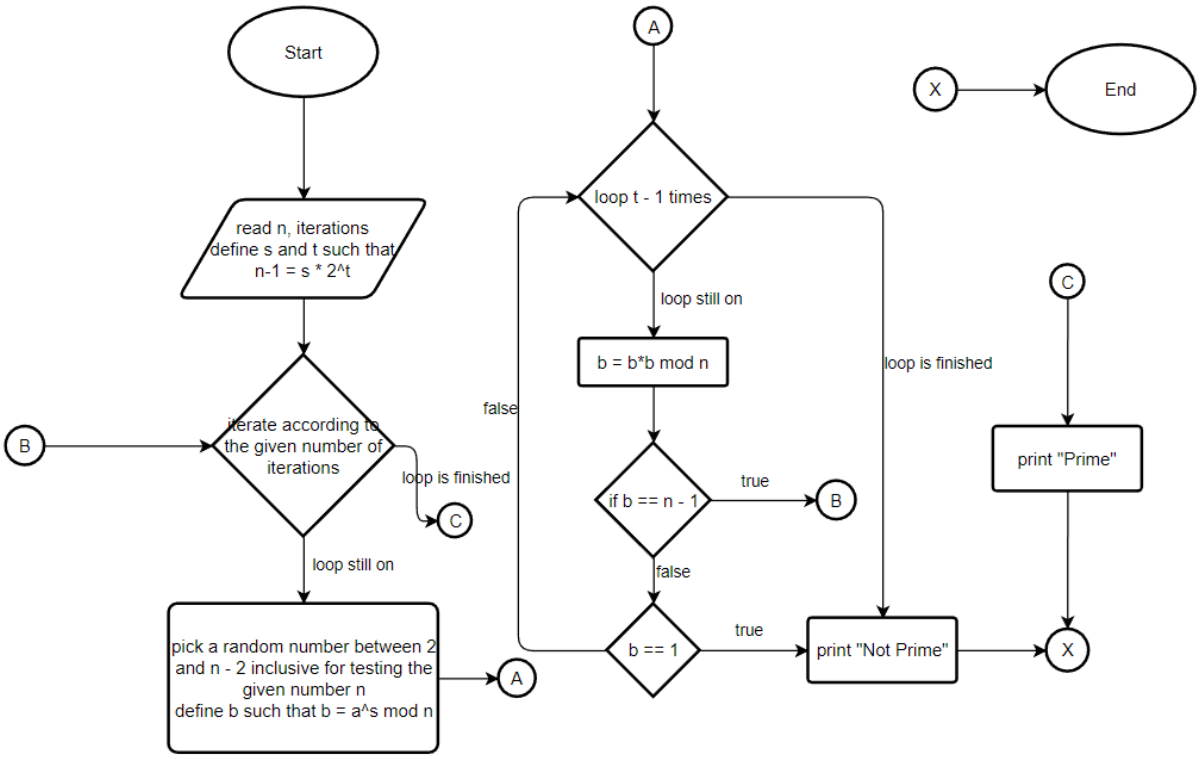
First: Problem Statement

It is required to implement Miller's test (a probabilistic primality test) to find whether a given integer is a prime or not.

Second: Used Data Structures

None.

Third: Flow Charts



Fourth: Assumptions

We assumed the user will not enter malicious input and stick to the bounds of the algorithm.

Fifth: Design Decisions

We used Fermat's little theorem to find the variable b since the build in pow() method of the c language couldn't handle the large numbers that were given to it as parameters and gave us wrong results.

Fifth: Sample Runs

```
Enter the number to check whether it is prime or not: 53
Enter the number of desired iterations: 10
53 is prime

Process returned 0 (0x0)   execution time : 12.169 s
Press any key to continue.
```

```
Enter the number to check whether it is prime or not: 3571
Enter the number of desired iterations: 10
3571 is prime

Process returned 0 (0x0)   execution time : 39.876 s
Press any key to continue.
```

```
Enter the number to check whether it is prime or not: 3573
Enter the number of desired iterations: 10
3573 is not prime

Process returned 0 (0x0)   execution time : 14.380 s
Press any key to continue.
```