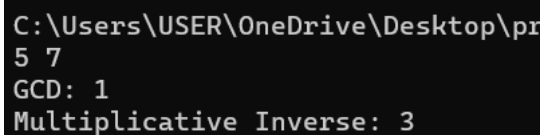


# Extended Euclidean - Implementation

## Code

```
import java.util.*;
public class ExtendedEuclidean {
    static int GCD(int num1, int num2) {
        if (num2 == 0)
            return num1;
        return GCD(num2, num1 % num2);
    }
    static int multiplicativeInverse(int num1, int num2, int t1, int t2) {
        if (num2 == 0)
            return t1;
        int t = t1 - (t2 * num1 / num2);
        return multiplicativeInverse(num2, num1 % num2, t2, t);
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int num1 = sc.nextInt();
        int num2 = sc.nextInt();
        int gcd = ExtendedEuclidean.GCD(num1, num2);
        System.out.println("GCD: " + gcd);
        if (gcd == 1) {
            if (num1 < num2)
                System.out.println("Multiplicative Inverse: " +
ExtendedEuclidean.multiplicativeInverse(num2, num1, 0, 1));
            else
                System.out.println("Multiplicative Inverse: " +
ExtendedEuclidean.multiplicativeInverse(num1, num2, 0, 1));
        }
    }
}
```

## Output



```
C:\Users\USER\OneDrive\Desktop\pr
5 7
GCD: 1
Multiplicative Inverse: 3
```

# Caesar's Cipher (Implementation)

## Code

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <ctype.h>

char *encrypt(char message[], int shift)
{
    int i;
    for (i = 0; i < strlen(message); i++)
    {
        // printf("Debug Original: %c = %d\n", message[i], message[i]);
        if ((int)message[i] == 32)
        {
            message[i] = message[i];
        }
        else if ((int)message[i] + shift > 126)
        {
            message[i] = (char)((((int)message[i] + shift) % 94));
        }
        else
        {
            message[i] = (char)((int)message[i] + shift);
        }
        // printf("Debug Encrypted: %c = %d\n", message[i], message[i]);
    }
    return message;
}

char *decrypt(char message[], int shift)
{
    int i;
    for (i = 0; i < strlen(message); i++)
    {
        // printf("Debug Original: %c = %d\n", message[i], message[i]);
        if ((int)message[i] == 32)
        {
            message[i] = message[i];
        }
        else if ((int)message[i] - shift < 32)
        {
            message[i] = (char)(126 - (32 - ((int)message[i] -
shift))));
        }
        else
        {
            message[i] = (char)((int)message[i] - shift);
        }
        // printf("Debug Decrypted: %c = %d\n", message[i], message[i]);
    }
    return message;
}

int main(int argc, char *argv[])
Dheeraj Lalwani
1902085
```

```

{
    if (strcmp(argv[1], "-e") == 0)
    {
        printf("You chose to encrypt data with a shift of %d\n",
atoi(argv[3]));
        printf("Original string: | %s |\n", argv[2]);
        printf("Encrypted string: | %s |\n", encrypt(argv[2],
atoi(argv[3])));
    }
    else if (strcmp(argv[1], "-d") == 0)
    {
        printf("You chose to decrypt data with a shift of %d\n",
atoi(argv[3]));
        printf("Original string: | %s |\n", argv[2]);
        printf("Decryped string: | %s |\n", decrypt(argv[2],
atoi(argv[3])));
    }
    else
    {
        printf("Please enter a valid option.");
    }

    return 0;
}

```

## Output

```

C:\Users\USER\OneDrive\Desktop\practicals\CSS\CesarCipher>gcc CaesarCipher.c -o CaesarCipher

C:\Users\USER\OneDrive\Desktop\practicals\CSS\CesarCipher>CaesarCipher -e "Is your name Nikita?" 15
You chose to encrypt data with a shift of 15
Original string: | Is your name Nikita? |
Encrypted string: | X$ *~&# }p|t ]xzx%pN |

C:\Users\USER\OneDrive\Desktop\practicals\CSS\CesarCipher>CaesarCipher -d "X$ *~&# }p|t ]xzx%pN" 15
You chose to decrypt data with a shift of 15
Original string: | X$ *~&# }p|t ]xzx%pN |
Decryped string: | Is your name Nikita? |

```

# Playfair Cipher - Implementation

## Code

```
#include<stdio.h>
#include<string.h>
#include<ctype.h>

#define MX 5

void playfair(char ch1, char ch2, char key[MX][MX]) {
    int i, j, w, x, y, z;
    FILE * out;
    if ((out = fopen("cipher.txt", "a+")) == NULL) {
        printf("File Corrupted.");
    }
    for (i = 0; i < MX; i++) {
        for (j = 0; j < MX; j++) {
            if (ch1 == key[i][j]) {
                w = i;
                x = j;
            } else if (ch2 == key[i][j]) {
                y = i;
                z = j;
            }
        }
    }
    if (w == y) {
        x = (x + 1) % 5;
        z = (z + 1) % 5;
        printf("%c%c", key[w][x], key[y][z]);
        fprintf(out, "%c%c", key[w][x], key[y][z]);
    } else if (x == z) {
        w = (w + 1) % 5;
        y = (y + 1) % 5;
        printf("%c%c", key[w][x], key[y][z]);
        fprintf(out, "%c%c", key[w][x], key[y][z]);
    } else {
```

```

        printf("%c%c", key[w][z], key[y][x]);
        fprintf(out, "%c%c", key[w][z], key[y][x]);
    }
    fclose(out);
}

void main() {
    int i, j, k = 0, l, m = 0, n;

    char key[MX][MX], keyminus[25], keystr[10], str[25] = {
        0
    };

    char alpa[26] = {'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J',
        'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y',
        'Z'};

    printf("\nEnter key: ");
    gets(keystr);

    printf("\nEnter the plain text: ");
    gets(str);

    n = strlen(keystr);

    //convert the characters of key to uppertext
    for (i = 0; i < n; i++) {
        if (keystr[i] == 'j') keystr[i] = 'i';
        else if (keystr[i] == 'J') keystr[i] = 'I';
        keystr[i] = toupper(keystr[i]);
    }

    //convert all the characters of plaintext message to uppertext
    for (i = 0; i < strlen(str); i++) {
        if (str[i] == 'j') str[i] = 'i';
        else if (str[i] == 'J') str[i] = 'I';
        str[i] = toupper(str[i]);
    }

```

```

}

j = 0;

for (i = 0; i < 26; i++) {
    for (k = 0; k < n; k++) {
        if (keyststr[k] == alpa[i])
            break;
        else if (alpa[i] == 'J')
            break;
    }
    if (k == n) {
        keyminus[j] = alpa[i];
        j++;
    }
}

//construct key keymatrix
k = 0;
for (i = 0; i < MX; i++) {
    for (j = 0; j < MX; j++) {
        if (k < n) {
            key[i][j] = keyststr[k];
            k++;
        } else {
            key[i][j] = keyminus[m];
            m++;
        }
        printf("%c ", key[i][j]);
    }
    printf("\n");
}

printf("\nEntered text: %s\nCipher Text: ", str);
for (i = 0; i < strlen(str); i++) {
    if (str[i] == 'J') str[i] = 'I';
    if (str[i + 1] == '\\0')
        playfair(str[i], 'X', key);
}

```

```

else {
    if (str[i + 1] == 'J') str[i + 1] = 'I';
    if (str[i] == str[i + 1])
        playfair(str[i], 'X', key);
    else {
        playfair(str[i], str[i + 1], key);
        i++;
    }
}
}
printf("\n");
}

```

## Output

```

C:\Users\USER\OneDrive\Desktop\practicals\CSS\CesarCipher>gcc CaesarCipher.c -o CaesarCipher

C:\Users\USER\OneDrive\Desktop\practicals\CSS\CesarCipher>CaesarCipher -e "Is your name Nikita?" 15
You chose to encrypt data with a shift of 15
Original string: | Is your name Nikita? |
Encrypted string: | X$ *~&# }p|t ]xzx%pN |

C:\Users\USER\OneDrive\Desktop\practicals\CSS\CesarCipher>CaesarCipher -d "X$ *~&# }p|t ]xzx%pN" 15
You chose to decrypt data with a shift of 15
Original string: | X$ *~&# }p|t ]xzx%pN |
Decryped string: | Is your name Nikita? |

```

# Euler's Totient - Implementation

## Code

```
public class EulerTotient
{
    static int gcd (int x, int y)
    {
        if(x == 0)
        {
            return y;
        }
        else
        {
            return gcd(y % x, x);
        }
    }
    static int phi(int n)
    {
        int result = 1;
        for (int i = 2; i < n; i++)
        {
            if(gcd(i, n) == 1)
            {
                result++;
            }
        }
        return result;
    }
    public static void main(String [] args)
    {
        int n;
        for(n = 1; n < 12; n++)
        {
            System.out.println("Phi(" + n + ") = " + phi(n));
        }
    }
}
```



## Output

```
C:\Users\USER\OneDrive\Desktop\practicals\CSS\EulerTotient>java EulerTotient
Phi(1) = 1
Phi(2) = 1
Phi(3) = 2
Phi(4) = 2
Phi(5) = 4
Phi(6) = 2
Phi(7) = 6
Phi(8) = 4
Phi(9) = 6
Phi(10) = 4
Phi(11) = 10
```

# RSA - Implementation

## Code

```
import java.math.BigInteger;
import java.util.*;
public class RSA {
    static int gcd(int x, int y) {
        if(x == 0) {
            return y;
        }
        else {
            return gcd(y % x, x);
        }
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int prime1, prime2, product, phi, publicKey, privateKey = 0,
        plainTextInt, cipherTextInt, decryptedTextInt;
        System.out.println("Implementing RSA algorithm: ");
        System.out.println("Let us generate the keys...");
        System.out.print("Choose 2 prime numbers: ");
        prime1 = sc.nextInt();
        prime2 = sc.nextInt();
        product = prime1 * prime2;
        phi = (prime1 - 1) * (prime2 - 1);
        System.out.println("Phi was calculated to be: " + phi);

        for(publicKey = 2; publicKey < phi; publicKey++) {
            if(gcd(publicKey, phi) == 1) {
                break;
            }
        }

        System.out.println("Public key was calculated to be: (" +
        publicKey + ", " + product + ")");

        for(int i = 0; i <= 9; i++) {
```

```

        int x = 1 + (i * phi);

        if(x % publicKey == 0) {
            privateKey = x / publicKey;
            break;
        }
    }

    System.out.println("Private key was calculated to be: (" +
privateKey + ", " + product + ")");
    System.out.print("Enter number to be encrypted: ");
    plainTextInt = sc.nextInt();
    cipherTextInt = (int)Math.pow(plainTextInt, publicKey);
    System.out.println("Encrypted message is: " + cipherTextInt);
    BigInteger PRODUCT = BigInteger.valueOf(product);
    BigInteger CIPHERTEXTINT = BigInteger.valueOf(cipherTextInt);
    BigInteger DECRYPTEDTEXTINT;
    DECRYPTEDTEXTINT = (CIPHERTEXTINT.pow(privateKey)).mod(PRODUCT);
    decryptedTextInt = DECRYPTEDTEXTINT.intValue();

    if(plainTextInt == decryptedTextInt) {
        System.out.println("Decrypted text is: " + decryptedTextInt);
    }
    else {
        System.out.println("Kuch to gadbad hai bhai...");
    }
}
}

```

## Output

```

C:\Users\USER\OneDrive\Desktop\practicals\CSS\RSA>java RSA
Implementing RSA algorithm:
Let us generate the keys...
Choose 2 prime numbers: 3 11
Phi was calculated to be: 20
Public key was calculated to be: (3, 33)
Private key was calculated to be: (7, 33)
Enter number to be encrypted: 12
Encrypted message is: 1728
Decrypted text is: 12

```