PR.NO-5TH

```
#include<iostream>
#include<stdlib.h>
#include<math.h>
#include<string.h>
using namespace std;
int x, y, -n, t, i, flag;
long int e[50], d[50], temp[50], j;
char en[50], m[50];
char msg[100];
int prime(long int); //function to check for prime number
void encryption_key();
long int cd(long int);
void encrypt();
void decrypt();
int main()
{
 cout << "\nENTER FIRST PRIME NUMBER\n";
 cin >> x;
 //checking whether input is prime or not
 flag = prime(x);
 if(flag == 0)
 {
 cout << "\n INVALID INPUT\n";
 exit(0);
 }
```

```
cout << "\nENTER SECOND PRIME NUMBER\n";
cin >> y;
flag = prime(y);
if(flag == 0 || x == y)
{
cout << "\nINVALID INPUT\n";
exit(0);
}
cout << "\nENTER MESSAGE OR STRING TO ENCRYPT\n";
cin >> msg;
for(i = 0; msg[i] != NULL; i++)
m[i] = msg[i];
n = x * y;
t = (x - 1) * (y - 1);
encryption_key();
cout << "\nPOSSIBLE VALUES OF e AND d ARE\n";
for(i = 0; i < j - 1; i++)
cout << "\n" << e[i] << "\t" << d[i];
encrypt();
decrypt();
return 0;
} //end of the main program
int prime(long int pr)
{
int i;
j = sqrt(pr);
```

```
 for(i = 2; i <= j; i++)
 {
 if(pr % i == 0)
 return 0;
 }
 return 1;
}
//function to generate encryption key
void encryption_key()
{
 int k;
 k = 0;
 for(i = 2; i < t; i++)
 {
 if(t % i == 0)
 continue;
 flag = prime(i);
 if(flag == 1 && i != x && i != y)
 {
 e[k] = i;
 flag = cd(e[k]);
 if(flag > 0)
 {
 d[k] = flag;
 k++;
 }
```

```
 if(k == 99)
 break;
 }
 }
}
long int cd(long int a)
{
 long int k = 1;
 while(1)
 {
 k = k + t;
 if(k % a == 0)
 return(k/a);
 }
}
//function to encrypt the message
void encrypt()
{
 long int pt, ct, key = e[0], k, len;
 i = 0;
 len = strlen(msg);
 while(i != len)
 {
 pt = m[i];
 pt = pt - 96;
 k = 1;
```

```
 for(j = 0; j < key; j++)
 {
 k = k * pt;
 k = k % n;
 }
 temp[i] = k;
 ct= k + 96;
 en[i] = ct;
 i++;
 }
 en[i] = -1;
 cout << "\n\nTHE ENCRYPTED MESSAGE IS\n";
 for(i=0; en[i] != -1; i++)
 cout << en[i];
}
//function to decrypt the message
void decrypt()
{
 long int pt, ct, key = d[0], k;
 i = 0;
 while(en[i] != -1)
 {
 ct = temp[i];
 k = 1;
 for(j = 0; j < key; j++)
 {
```

```
k = k * ct;
k = k % n;
}
pt = k + 96;
m[i] = pt;
i++;
}
m[i] = -1;
cout << "\n\nTHE DECRYPTED MESSAGE IS\n";
for(i = 0; m[i] != -1; i++)
cout << m[i];
cout << endl;
}
```

PROGRAM EXPLANATION –

```
#include<iostream>
#include<stdlib.h>
#include<math.h>
#include<string.h>
using namespace std;
```

These are the include statements that include the necessary header files for input/output, mathematical operations, string handling, and the standard namespace.

```
int x, y, n, t, i, flag;
long int e[50], d[50], temp[50], j;
char en[50], m[50];
char msg[100];
```

Here, variables are declared to hold various values used in the program. `x` and `y` are prime numbers input by the user. `n` is the product of `x` and `y`. `t` is used for some calculations based on `x` and `y`. `e` and `d` are arrays to store encryption and decryption keys. `temp` is an array used for intermediate calculations. `j` is a counter variable. `en` is an array to store the encrypted message, and `m` is an array to store the decrypted message. `msg` is a character array to store the input message.

```
int prime(long int); //function to check for prime number
void encryption_key();
long int cd(long int);
void encrypt();
void decrypt();
```

These are function prototypes. They indicate the presence of functions that will be defined later in the program.

```
int main()
{
  // ...
}
```

This is the main function, where the execution of the program starts.

```
cout << "\nENTER FIRST PRIME NUMBER\n";
cin >> x;
flag = prime(x);
if(flag == 0)
{
  cout << "\nINVALID INPUT\n";
  exit(0);
}
```

The program prompts the user to enter the first prime number (`x`) and checks if it is indeed a prime number. If it's not prime, it displays an error message and terminates the program.

```
cout << "\nENTER SECOND PRIME NUMBER\n";
cin >> y;
flag = prime(y);
if(flag == 0 || x == y)
{
  cout << "\n INVALID INPUT\n";
  exit(0);
}
```

Similarly, the program prompts the user to enter the second prime number (`y`). It checks if it is prime and also checks if it is the same as `x`. If either condition is true, it displays an error message and terminates the program

```
cout << "\nENTER MESSAGE OR STRING TO ENCRYPT\n";
cin >> msg;
for(i = 0; msg[i] != NULL; i++)
  m[i] = msg[i];
```

The program prompts the user to enter a message or string to encrypt. It reads the input and copies it character by character into the array `m`.

```
n = x * y;
t = (x - 1) * (y - 1);
encryption_key();
```

The program calculates the value of `n` by multiplying `x` and `y`, and the value of `t` by multiplying `x-1` and `y-1`. Then, it calls the `encryption_key()` function to generate the possible encryption and decryption keys.

```
cout << "\nPOSSIBLE VALUES OF e AND d ARE\n";
for(i = 0; i < j - 1; i++)
  cout << "\n" << e[i] << "\t" << d[i];
```

After generating the encryption and decryption keys, the program displays the possible values of `e` and `d` on separate lines.

```
encrypt();
decrypt();
```

```
int prime(long int pr)
{
 int i;
 j = sqrt(pr);
 for(i = 2; i <= j; i++)
 {
 if(pr % i == 0)
 return 0;
 }
 return 1;
}
//function to generate encryption key
void encryption_key()
{
 int k;
 k = 0;
 for(i = 2; i < t; i++)
 {
 if(t % i == 0)
 continue;
 flag = prime(i);
 if(flag == 1 && i != x && i != y)
 {
 e[k] = i;
 flag = cd(e[k]);
 if(flag > 0)
```

```
 {
 d[k] = flag;
 k++;
 }
 if(k == 99)
 break;
 }
 }
}
long int cd(long int a)
{
 long int k = 1;
 while(1)
 {
 k = k + t;
 if(k % a == 0)
 return(k/a);
 }
}
//function to encrypt the message
void encrypt()
{
 long int pt, ct, key = e[0], k, len;
 i = 0;
 len = strlen(msg);
 while(i != len)
```

```
{
pt = m[i];
pt = pt - 96;
k = 1;
for(j = 0; j < key; j++)
{
k = k * pt;
k = k % n;
}
temp[i] = k;
ct= k + 96;
en[i] = ct;
i++;
}
en[i] = -1;
cout << "\n\nTHE ENCRYPTED MESSAGE IS\n";
for(i=0; en[i] != -1; i++)
cout << en[i];
}
//function to decrypt the message
void decrypt()
{
long int pt, ct, key = d[0], k;
i = 0;
while(en[i] != -1)
{
```

```
ct = temp[i];
k = 1;
for(j = 0; j < key; j++)
{
k = k * ct;
k = k % n;
}
pt = k + 96;
m[i] = pt;
i++;
}
m[i] = -1;
cout << "\n\nTHE DECRYPTED MESSAGE IS\n";
for(i = 0; m[i] != -1; i++)
cout << m[i];
cout << endl;
}
```

PROGRAM EXPLAIN –

int prime(long int pr): This line defines a function named prime that takes a long int parameter pr and returns an int.

int i;: This line declares a variable i of type int.

j = sqrt(pr);: This line calculates the square root of pr and assigns it to the variable j.

for(i = 2; i <= j; i++): This line starts a for loop that iterates from 2 to j, incrementing i by 1 in each iteration.

if(pr % i == 0): This line checks if pr is divisible evenly by i.

return 0;: This line returns 0, indicating that pr is not a prime number.

return 1;: This line returns 1, indicating that pr is a prime number.

void encryption_key(): This line defines a function named encryption_key that returns void (no return value).

int k;: This line declares a variable k of type int.

k = 0;: This line initializes k to 0.

for(i = 2; i < t; i++): This line starts a for loop that iterates from 2 to t-1, incrementing i by 1 in each iteration.

if(t % i == 0) continue;: This line checks if t is divisible evenly by i and continues to the next iteration if true.

flag = prime(i);: This line calls the prime function to check if i is a prime number and assigns the result to flag.

if(flag == 1 && i != x && i != y): This line checks if flag is 1 (indicating that i is a prime number) and i is not equal to x or y.

e[k] = i;: This line assigns the value of i to the k-th element of the array e.

flag = cd(e[k]);: This line calls the cd function with e[k] as the argument and assigns the result to flag.

if(flag > 0): This line checks if flag is greater than 0.

d[k] = flag;: This line assigns the value of flag to the k-th element of the array d.

k++;: This line increments k by 1.

if(k == 99) break; : This line checks if k is equal to 99 and breaks out of the loop if true.

long int cd(long int a): This line defines a function named cd that takes a long int parameter a and returns a long int.

long int k = 1; : This line declares a variable k of type long int and initializes it to 1.

while (1): This line starts an infinite loop.

k = k + t; : This line increments k by t.

if(k % a == 0) return(k/a);: This line checks if k is divisible evenly by a and returns the result of the division if true.

void encrypt (): This line

PR.NO-6TH

```
"//ceaser cipher
#include<iostream>
#include<string.h>
using namespace std;
stringa encrypt(string plain_text, int key)
{
string cipher_text="";
for(int i=0;i< plain_text.size(); i++)
{
if(isupper(plain_text[i]))
{
cipher_text+= char(int(plain_text[i]+ key- 65)%26+65);
}
else
{
cipher_text+= char(int(plain_text[i]+ key- 97)%26+ 97);
}
}
return cipher_text;
}
string decrypt(string cipher_text, int key)
{
string plain_text="";
for(int i=0;i< cipher_text.size(); i++)
{
```

```
if(isupper(cipher_text[i]))
{
plain_text+= char((int(cipher_text[i]- key- 65)%26+26)%26 + 65);
}

else
{
plain_text+= char((int(cipher_text[i]- key- 97)%26+ 26)%26 + 97);
}
}
return plain_text;
}
int main()
{
string plain_text;
cout<<"Enter plain text: ";
cin>>plain_text;
int key;
cout<<"Enter key: ";
cin>>key;
string cipher_text= encrypt(plain_text, key);
cout<<"\nEncryption: \n";
cout<<"Cipher: "<<cipher_text<<endl;
cout<<"\nDecryption:\n";
cout<<"Plain Text: "<<decrypt(cipher_text, key)<<endl
}"
```

PROGRAM EXPLANATION –

```
//ceaser cipher
#include<iostream>
#include<string.h>
```

These lines are comments and include necessary header files for input/output and string manipulation.

```
using namespace std;
```

This line indicates that we're using the standard namespace, which allows us to use standard library functions without specifying their namespace.

```
string encrypt(string plain_text, int key)
{
   string cipher_text = "";
   for(int i = 0; i < plain_text.size(); i++)
   {
      if(isupper(plain_text[i]))
      {
         cipher_text += char(int(plain_text[i] + key - 65) % 26 + 65);
      }
      else
      {
         cipher_text += char(int(plain_text[i] + key - 97) % 26 + 97);
      }
   }
   return cipher_text;
}
```

- This function `encrypt` takes a `plain_text` string and an integer `key` as parameters and returns the encrypted text. It initializes an empty string `cipher_text`.
- The function loops through each character of the `plain_text` using `i` as the index.
- If the character is an uppercase letter, it performs the Caesar cipher encryption by adding the key value to the ASCII value of the character and applies modulo operation to ensure it stays within the range of uppercase letters. It then converts the resulting ASCII value back to a character and appends it to `cipher_text`.
- If the character is a lowercase letter, it performs the same process but using the ASCII range of lowercase letters.
- Once all characters have been processed, it returns the `cipher_text`.

```
string decrypt(string cipher_text, int key)
{
   string plain_text = "";
   for(int i = 0; i < cipher_text.size(); i++)
   {
      if(isupper(cipher_text[i]))
      {
         plain_text += char((int(cipher_text[i] - key - 65) % 26 + 26) % 26 + 65);
      }
      else
      {
         plain_text += char((int(cipher_text[i] - key - 97) % 26 + 26) % 26 + 97);
      }
   }
   return plain_text;
}
```

- This function `decrypt` takes a `cipher_text` string and an integer `key` as parameters and returns the decrypted text. It initializes an empty string `plain_text`.
- The function loops through each character of the `cipher_text` using `i` as the index.
- If the character is an uppercase letter, it performs the reverse Caesar cipher decryption by subtracting the key value from the ASCII value of the character. It then applies the modulo operation to handle negative values and converts the resulting ASCII value back to a character, which is appended to `plain_text`.
- If the character is a lowercase letter, it performs the same process but using the ASCII range of lowercase letters.
- Once all characters have been processed, it returns the `plain_text`.

```
int main()
{
    string plain_text;
    cout << "Enter plain text: ";
    cin >> plain_text;
    int key;
    cout << "Enter key: ";
    cin >> key;
    string cipher_text = encrypt(plain_text, key);
    cout << "\nEncryption: \n";
    cout << "Cipher: " << cipher_text << endl;
    cout << "\nDecryption:\n";
    cout << "Plain Text: " << decrypt(cipher_text, key) << endl;
}
```

Explanation:-

int main()

This is the entry point of the program. It declares the `main` function, which is required in every C++ program.

{

```
string plain_text;
cout << "Enter plain text: ";
cin >> plain_text;
```

- Declares a string variable `plain_text` to store the user's input for the plain text.
- Displays the prompt "Enter plain text: " to instruct the user to enter their desired plain text.
- Reads the user's input from the standard input using `cin` and stores it in `plain_text`.

```
int key;
cout << "Enter key: ";
cin >> key;
```

- Declares an integer variable `key` to store the user's input for the encryption key.
- Displays the prompt "Enter key: " to instruct the user to enter the encryption key.
- Reads the user's input from the standard input using `cin` and stores it in `key`.

```
string cipher_text = encrypt(plain_text, key);
cout << "\nEncryption: \n";
cout << "Cipher: " << cipher_text << endl;
```

- Declares a string variable `cipher_text` and assigns the result of calling the `encrypt` function with `plain_text` and `key` as arguments.
- Displays the header "Encryption:" to indicate the following output is the result of encryption.

- Prints the encrypted text by concatenating the string "Cipher: " and `cipher_text` using the `<<` operator.
- Ends the line with `endl` to move to the next line.

```
cout << "\nDecryption:\n";
cout << "Plain Text: " << decrypt(cipher_text, key) << endl;
```

- Displays the header "Decryption:" to indicate the following output is the result of decryption.
- Prints the decrypted text by concatenating the string "Plain Text: " and the result of calling the `decrypt` function with `cipher_text` and `key` as arguments.
- Ends the line with `endl` to move to the next line.

```
}
```

- Closes the `main` function and marks the end of the program.

PR.NO-6TH

```
#include<iostream>
using namespace std;
class LoginManger
{
	public:
		string userNameAttempt;
		string passWordAttempt;

		void login()
		{
			cout<<"hey you need to enter your USENAME and PASSWORD:";
			cin>>userNameAttempt;
			if(userNameAttempt==userName)
			{
				cout<<"password:";
				cin>>passWordAttempt;
				if(passWordAttempt==passWord)

				{
					cout<<"Hey,thats rightr:";
				}
			}
		}

		private:
			string passWord="gsm101";
			string userName="gsmcoe@gmail";
};

int main ()
{
	LoginManger loginMangerObj;
	loginMangerObj.login();
}
```

EXPLAINS –

#include<iostream>

using namespace std;

These lines include the necessary header file `iostream` for input/output operations and declare the usage of the `std` namespace.

```
class LoginManager
{
    public:
        string userNameAttempt;
        string passWordAttempt;

        void login()
        {
            cout << "Hey, you need to enter your USERNAME and PASSWORD: ";
            cin >> userNameAttempt;
            if(userNameAttempt == userName)
            {
                cout << "Password: ";
                cin >> passWordAttempt;
                if(passWordAttempt == passWord)
                {
                    cout << "Hey, that's right!";
                }
            }
        }

    private:
        string passWord = "kusal";
        string userName = "a@gmail";
};
```

- This code defines a class named `LoginManager` which encapsulates the functionality for user login.
- The class contains two public member variables `userNameAttempt` and `passWordAttempt`, which will store the user's input for the username and password attempts, respectively.
- The class also includes a public member function named `login` which implements the login process.
- Within the `login` function, it prompts the user to enter their username and reads their input into `userNameAttempt` using `cin`.
- It then checks if the entered `userNameAttempt` matches the stored `userName`.
- If the username is correct, it prompts the user to enter their password and reads their input into `passWordAttempt` using `cin`.
- It then checks if the entered `passWordAttempt` matches the stored `passWord`.
- If both the username and password are correct, it displays the message "Hey, that's right!" using `cout`.

```
int main()
{
   LoginManager loginManagerObj;
   loginManagerObj.login();
}
```

- This is the `main` function where the program execution starts.
- It creates an instance of the `LoginManager` class named `loginManagerObj`.
- It then calls the `login` member function of the `loginManagerObj` object to start the login process.

Overall, the program allows the user to enter a username and password attempt and checks if they match the predefined username and password. If both match, it displays a success message.