SOLUTION BOOK

❤️ **From SIDDHARTH SINGH**

FUNCTION

1) **Program to Display Prime Numbers Between Two Intervals Using Functions**

CODE:

```
#include <iostream>
using namespace std;

int checkPrimeNumber(int);

int main() {
    int n1, n2;
    bool flag;

    cout << "Enter two positive integers: ";
    cin >> n1 >> n2;

    // swapping n1 and n2 if n1 is greater than n2
    if (n1 > n2) {
      n2 = n1 + n2;
      n1 = n2 - n1;
      n2 = n2 - n1;
    }

    cout << "Prime numbers between " << n1 << " and " << n2 << " are: ";

    for(int i = n1+1; i < n2; ++i) {
        // If i is a prime number, flag will be equal to 1
        flag = checkPrimeNumber(i);

        if(flag)
            cout << i << " ";
    }

    return 0;
}

// user-defined function to check prime number
```

```
int checkPrimeNumber(int n) {
    bool isPrime = true;

    // 0 and 1 are not prime numbers
    if (n == 0 || n == 1) {
        isPrime = false;
    }
    else {
        for(int j = 2; j <= n/2; ++j) {
            if (n%j == 0) {
                isPrime = false;
                break;
            }
        }
    }

    return isPrime;
}
```

Output

```
Enter two positive integers: 12
55
Prime numbers between 12 and 55 are: 13 17 19 23 29 31 37 41 43 47 53
```

2) Program to Check Whether a Number can be Express as Sum of Two Prime Numbers

CODE:

```
#include <iostream>
using namespace std;

bool checkPrime(int n);

int main() {
    int n, i;
    bool flag = false;

    cout << "Enter a positive  integer: ";
    cin >> n;

    for(i = 2; i <= n/2; ++i) {
        if (checkPrime(i)) {
            if (checkPrime(n - i)) {
                cout << n << " = " << i << " + " << n-i << endl;
                flag = true;
            }
        }
    }

    if (!flag)
      cout << n << " can't be expressed as sum of two prime numbers.";

    return 0;
}

// Check prime number
bool checkPrime(int n) {
    int i;
    bool isPrime = true;

    // 0 and 1 are not prime numbers
    if (n == 0 || n == 1) {
        isPrime = false;
    }
    else {
        for(i = 2; i <= n/2; ++i) {
            if(n % i == 0) {
                isPrime = false;
                break;
            }
        }
    }

    return isPrime;
}
```

Output

```
Enter a positive integer: 34
34 = 3 + 31
34 = 5 + 29
34 = 11 + 23
34 = 17 + 17
```

3) Program to Convert Binary Number to Decimal

CODE:

```
#include <iostream>
#include <cmath>

using namespace std;

int convertBinaryToDecimal(long long);

int main()
{
    long long n;

    cout << "Enter a binary number: ";
    cin >> n;

    cout << n << " in binary = " << convertBinaryToDecimal(n) << "in decimal";
    return 0;
}

int convertBinaryToDecimal(long long n)
{
    int decimalNumber = 0, i = 0, remainder;
    while (n!=0)
    {
        remainder = n%10;
        n /= 10;
        decimalNumber += remainder*pow(2,i);
        ++i;
    }
    return decimalNumber;
}
```

Output

```
Enter a binary number: 1111
1111 in binary = 15
```

4) Program to convert decimal number to binary

CODE:

```
#include <iostream>
#include <cmath>
```

```
using namespace std;

long long convertDecimalToBinary(int);

int main()
{
    int n, binaryNumber;

    cout << "Enter a decimal number: ";
    cin >> n;
    binaryNumber = convertDecimalToBinary(n);
    cout << n << " in decimal = " << binaryNumber << " in binary" << endl ;
    return 0;
}

long long convertDecimalToBinary(int n)
{
    long long binaryNumber = 0;
    int remainder, i = 1, step = 1;

    while (n!=0)
    {
        remainder = n%2;
        cout << "Step " << step++ << ": " << n << "/2, Remainder = " << remainder << ", Quotient = " << n/2 << endl;
        n /= 2;
        binaryNumber += remainder*i;
        i *= 10;
    }
    return binaryNumber;
}
```

Output

```
Enter a decimal number: 19
Step 1: 19/2, Remainder = 1, Quotient = 9
Step 2: 9/2, Remainder = 1, Quotient = 4
Step 3: 4/2, Remainder = 0, Quotient = 2
Step 4: 2/2, Remainder = 0, Quotient = 1
Step 5: 1/2, Remainder = 1, Quotient = 0
19 in decimal = 10011 in binary
```

5) **Program to Find Sum of Natural Numbers using Recursion**

CODE:

```
#include<iostream>
using namespace std;

int add(int n);

int main() {
    int n;

    cout << "Enter a positive integer: ";
    cin >> n;

    cout << "Sum =  " << add(n);

    return 0;
}

int add(int n) {
    if(n != 0)
        return n + add(n - 1);
    return 0;
}
```

Output

```
Enter an positive integer: 10
Sum = 55
```

6) **Program to Calculate Factorial of a Number Using Recursion**

CODE:

```
#include<iostream>
using namespace std;

int factorial(int n);

int main()
{
    int n;

    cout << "Enter a positive integer: ";
    cin >> n;

    cout << "Factorial of " << n << " = " << factorial(n);

    return 0;
}

int factorial(int n)
{
    if(n > 1)
        return n * factorial(n - 1);
    else
        return 1;
}
```

Output

```
Enter an positive integer: 6
Factorial of 6 = 720
```

7) Program to Find G.C.D Using Recursion

CODE:

```
#include <iostream>
using namespace std;

int hcf(int n1, int n2);

int main()
{
   int n1, n2;

   cout << "Enter two positive integers: ";
   cin >> n1 >> n2;

   cout << "H.C.F of " << n1 << " & " <<  n2 << " is: " << hcf(n1, n2);

   return 0;
}
```

```
int hcf(int n1, int n2)
{
    if (n2 != 0)
       return hcf(n2, n1 % n2);
    else
       return n1;
}
```

Output

```
Enter two positive integers: 366 60
H.C.F of 366 and 60 is: 6
```

CONCEPT:

Euclidean Algorithm is used here Refer

8) Program to Calculate Power Using Recursion

CODE:

```
#include <iostream>
using namespace std;

int calculatePower(int, int);

int main()
{
    int base, powerRaised, result;

    cout << "Enter base number: ";
    cin >> base;

    cout << "Enter power number(positive integer): ";
    cin >> powerRaised;

    result = calculatePower(base, powerRaised);
    cout << base << "^" << powerRaised << " = " << result;

    return 0;
}

int calculatePower(int base, int powerRaised)
{
    if (powerRaised != 0)
        return (base*calculatePower(base, powerRaised-1));
    else
```

```
        return 1;
}
```

Output

```
Enter base number: 3
Enter power number(positive integer): 4
3^4 = 81
```

NOTE:

This technique can only calculate power if the exponent is a positive integer.

To find power of any number, you can use **pow()** function.

result = pow(base, exponent);