UCS 2265 Fundamentals and Practice of Software Development

A3: Programs using User-Defined Functions Batch

2023-2027

Name: SK Shivaanee Academic Year 2023-2024 Even
Section: CSE B
Register ID: 31222300123

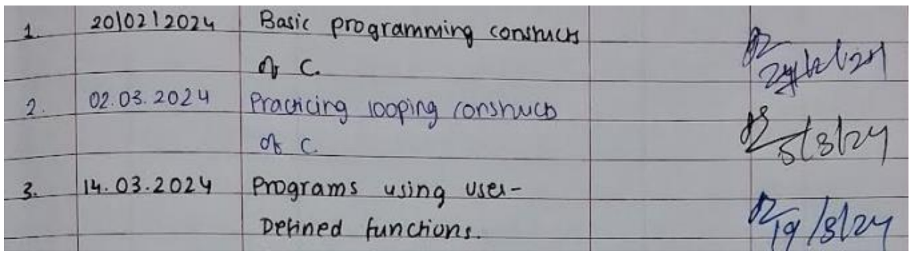
Dr. Chitra Babu, Dr. D. Thenmozhi, Dr. M. Saritha Learning

Outcome:

- You will be able to apply computational thinking to devise solutions through modular programming.
- You code C programs by defining the functions with appropriate parameters and return statements.

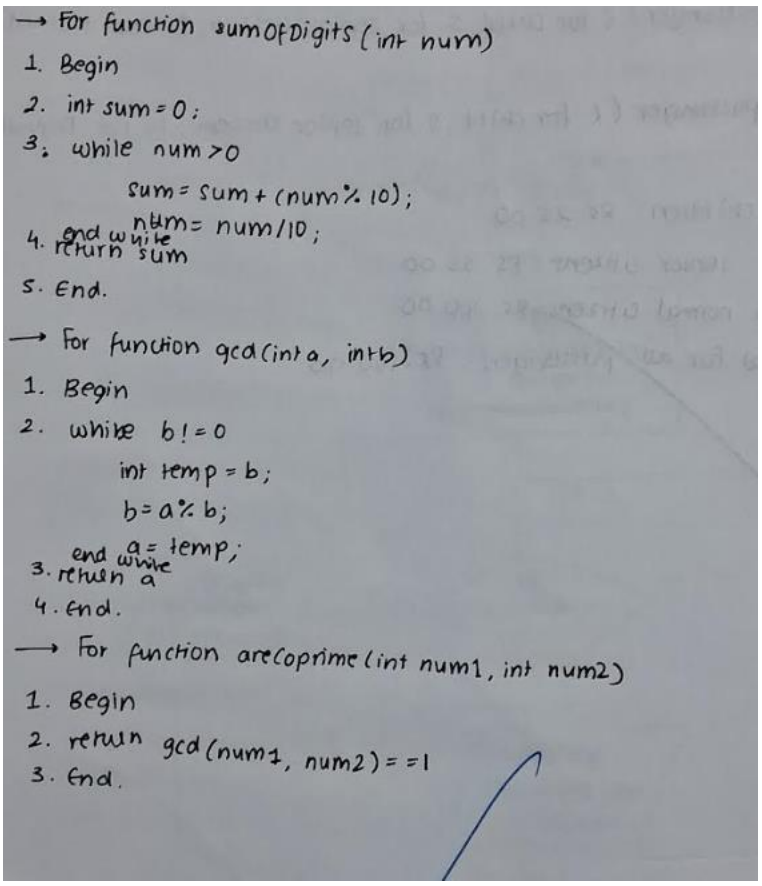
Best Practices: Naming conventions to be followed. Use comment statements. Avoid reading and printing statements within function definition.

Assignment: Solve the following problems by implementing in C. (CO7, K3, 1.3.1, 1.4.1, 2.1.2, 2.1.3, 4.2.1, 14.2.1, 14.2.2)



1.	20|02|2024	Basic programming constructs of C.		[illegible] 24/2/24
2.	02.03.2024	Practicing looping constructs of C.		[illegible] 5/3/24
3.	14.03.2024	Programs using user-Defined functions.		[illegible] 19/3/24

1. Apply modularity to solve the problem given in Assignment 2.4. (Finding the given pair of elements that are co-prime or not.)
   a. Identify the possible input and return value for each of the functions
   b. Write a pseudo code for each of the functions
   c. Develop a program to check for the co-primes and test it for any 4 pairs of numbers by calling the appropriate functions.



→ For function sumOfDigits (int num)

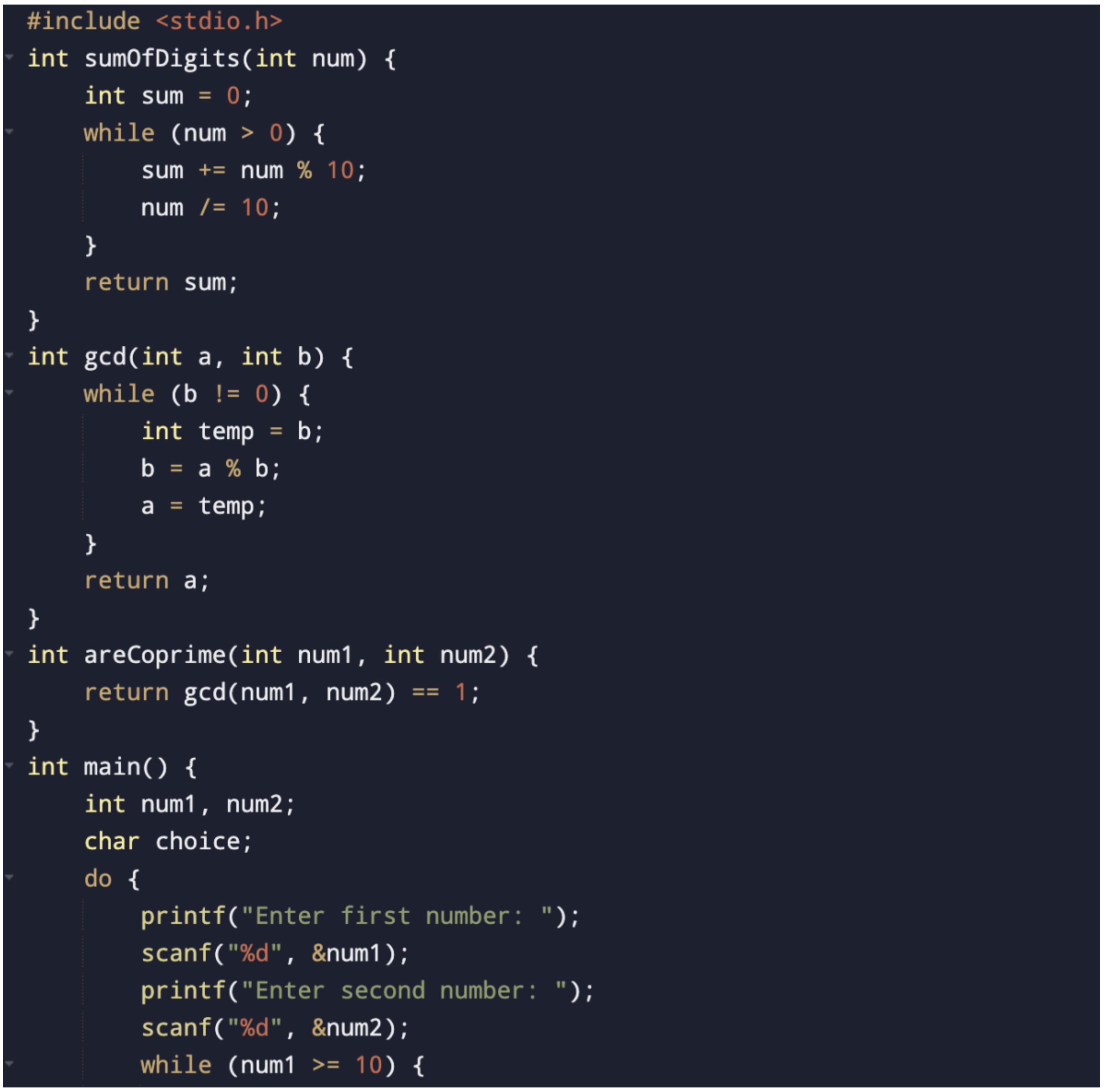
```
1. Begin
2. int sum = 0;
3. while num > 0
        sum = sum + (num % 10);
        num = num/10;
   end while
4. return sum
5. End.
```

→ For function gcd(int a, int b)

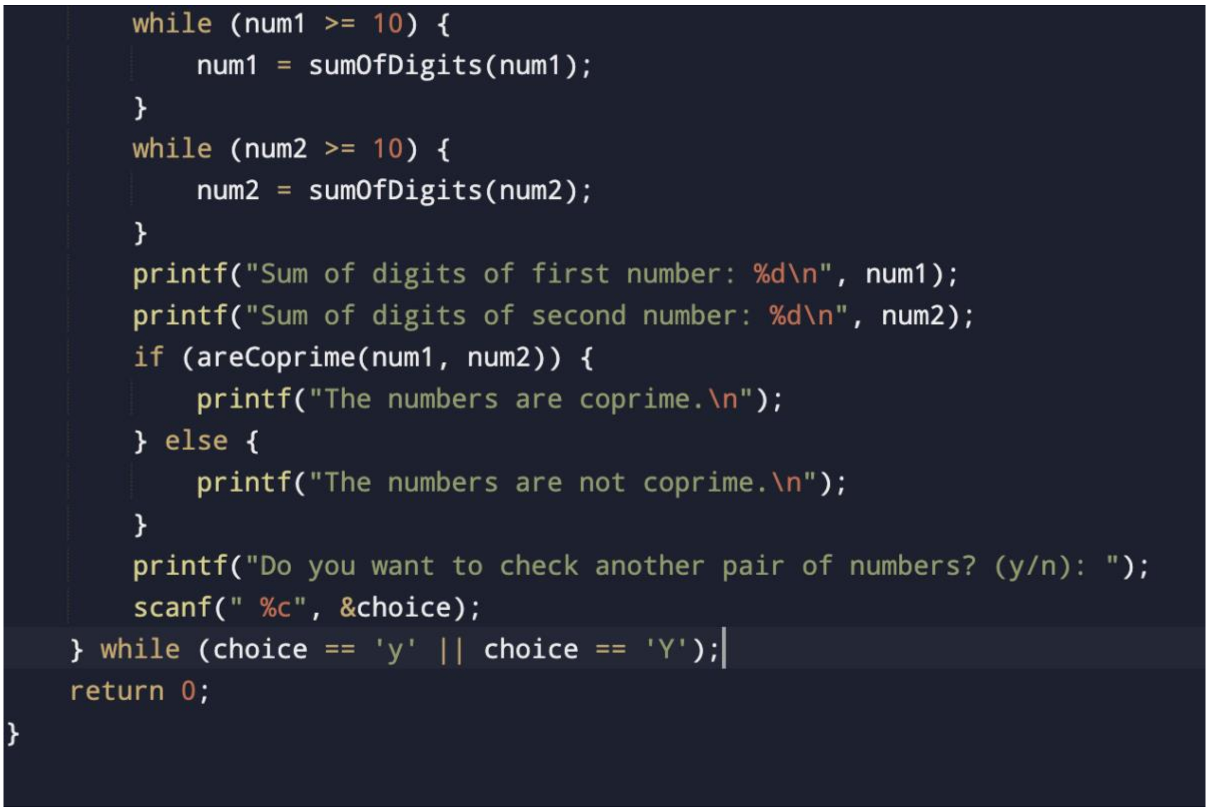
```
1. Begin
2. while b != 0
        int temp = b;
        b = a % b;
        a = temp;
   end while
3. return a
4. End.
```

→ For function areCoprime(int num1, int num2)

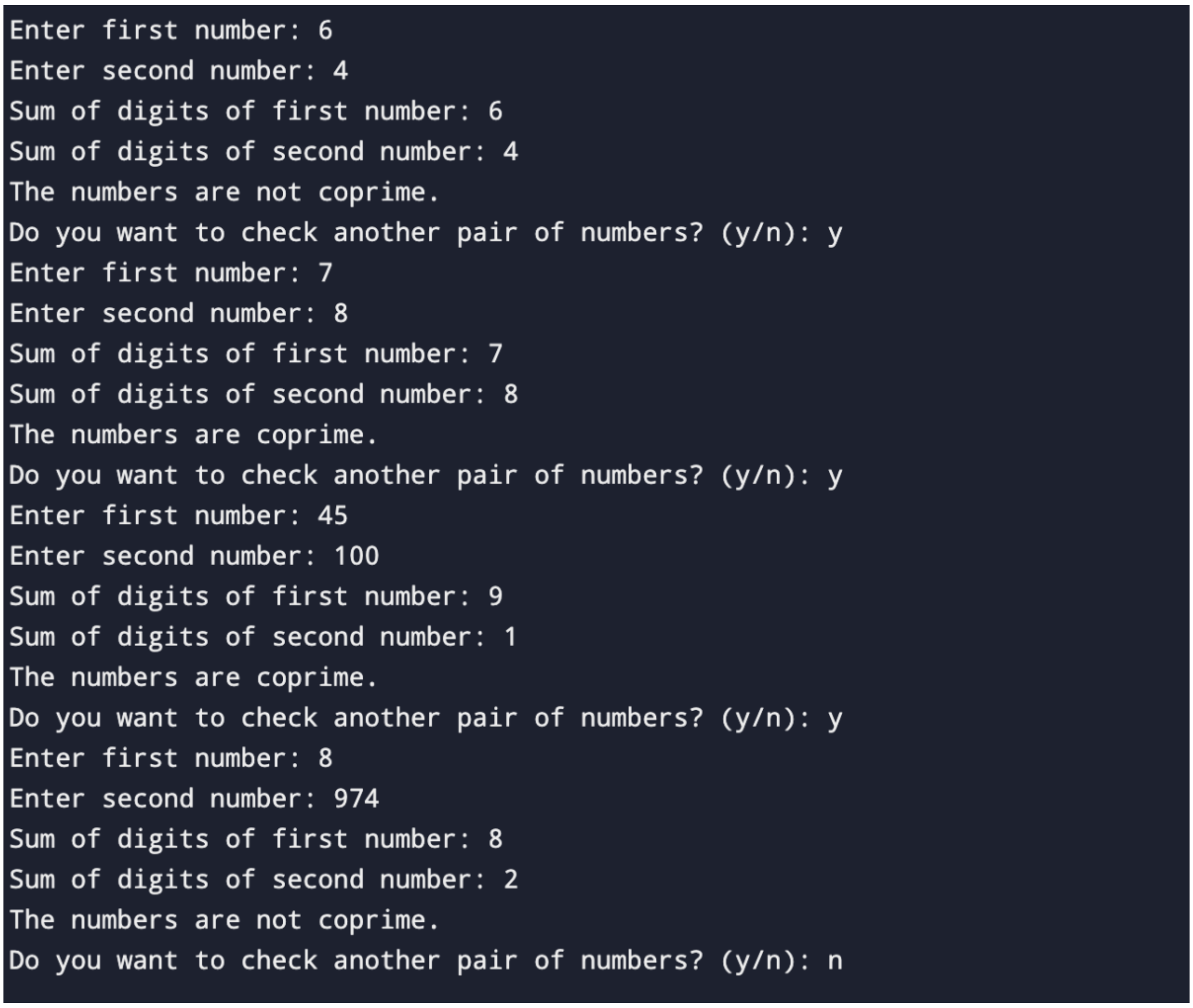
```
1. Begin
2. return gcd(num1, num2) == 1
3. End.
```



```c
#include <stdio.h>
int sumOfDigits(int num) {
    int sum = 0;
    while (num > 0) {
        sum += num % 10;
        num /= 10;
    }
    return sum;
}
int gcd(int a, int b) {
    while (b != 0) {
        int temp = b;
        b = a % b;
        a = temp;
    }
    return a;
}
int areCoprime(int num1, int num2) {
    return gcd(num1, num2) == 1;
}
int main() {
    int num1, num2;
    char choice;
    do {
        printf("Enter first number: ");
        scanf("%d", &num1);
        printf("Enter second number: ");
        scanf("%d", &num2);
        while (num1 >= 10) {
```



```c
        while (num1 >= 10) {
            num1 = sumOfDigits(num1);
        }
        while (num2 >= 10) {
            num2 = sumOfDigits(num2);
        }
        printf("Sum of digits of first number: %d\n", num1);
        printf("Sum of digits of second number: %d\n", num2);
        if (areCoprime(num1, num2)) {
            printf("The numbers are coprime.\n");
        } else {
            printf("The numbers are not coprime.\n");
        }
        printf("Do you want to check another pair of numbers? (y/n): ");
        scanf(" %c", &choice);
    } while (choice == 'y' || choice == 'Y');
    return 0;
}
```



```
Enter first number: 6
Enter second number: 4
Sum of digits of first number: 6
Sum of digits of second number: 4
The numbers are not coprime.
Do you want to check another pair of numbers? (y/n): y
Enter first number: 7
Enter second number: 8
Sum of digits of first number: 7
Sum of digits of second number: 8
The numbers are coprime.
Do you want to check another pair of numbers? (y/n): y
Enter first number: 45
Enter second number: 100
Sum of digits of first number: 9
Sum of digits of second number: 1
The numbers are coprime.
Do you want to check another pair of numbers? (y/n): y
Enter first number: 8
Enter second number: 974
Sum of digits of first number: 8
Sum of digits of second number: 2
The numbers are not coprime.
Do you want to check another pair of numbers? (y/n): n
```

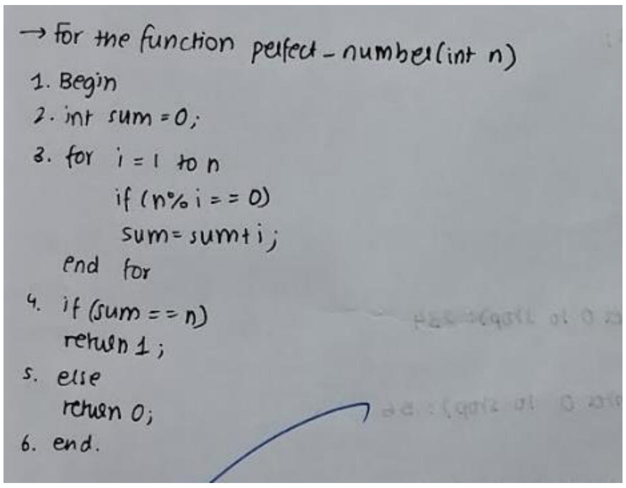
2.A perfect number is a positive integer that is equal to the sum of its positive divisors, excluding the number itself. Write a program in C to print all perfect numbers in a given range using a function to check whether a given number is a perfect number or not.

Example :

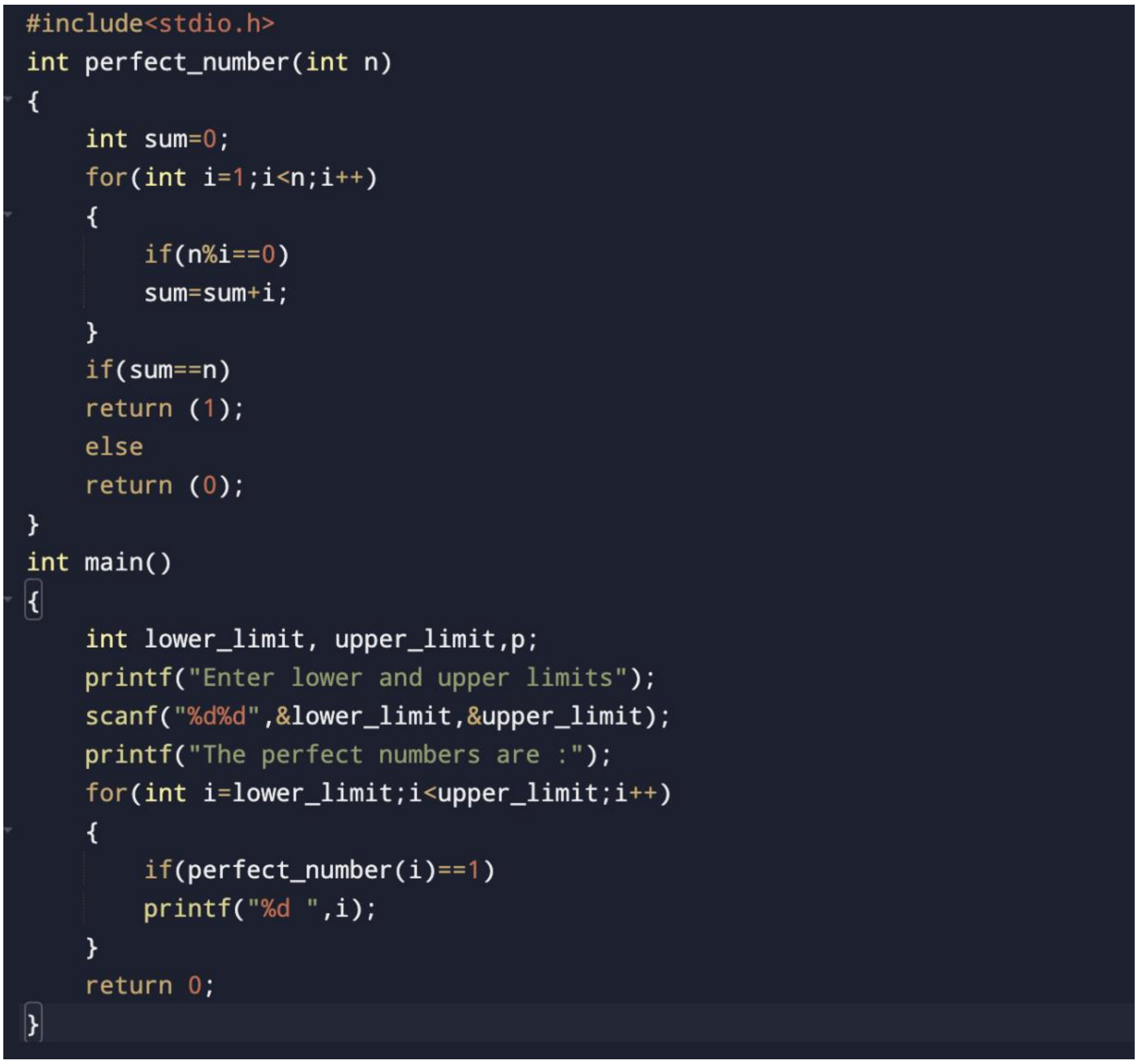
Input lower threshold : 1

Input higher threshold: 100

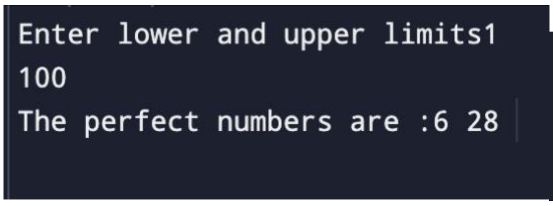
Expected Output : The perfect numbers between 1 to 100 are : 6 28



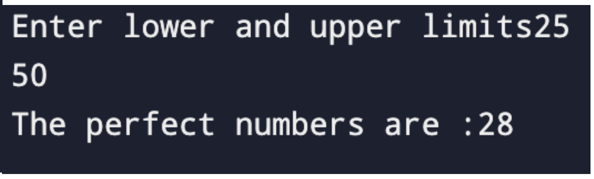
```
→ for the function perfect-number(int n)
1. Begin
2. int sum =0;
3. for i = 1 to n
         if (n% i = = 0)
         sum= sum+i;
   end for
4. if (sum = = n)
   return 1;
5. else
   return 0;
6. end.
```



```c
#include<stdio.h>
int perfect_number(int n)
{
    int sum=0;
    for(int i=1;i<n;i++)
    {
        if(n%i==0)
        sum=sum+i;
    }
    if(sum==n)
    return (1);
    else
    return (0);
}
int main()
{
    int lower_limit, upper_limit,p;
    printf("Enter lower and upper limits");
    scanf("%d%d",&lower_limit,&upper_limit);
    printf("The perfect numbers are :");
    for(int i=lower_limit;i<upper_limit;i++)
    {
        if(perfect_number(i)==1)
        printf("%d ",i);
    }
    return 0;
}
```

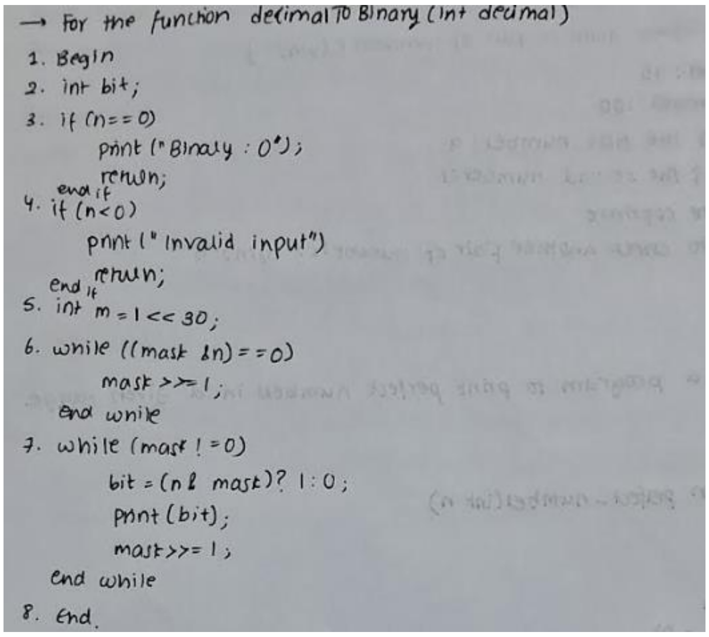


```
Enter lower and upper limits1
100
The perfect numbers are :6 28
```

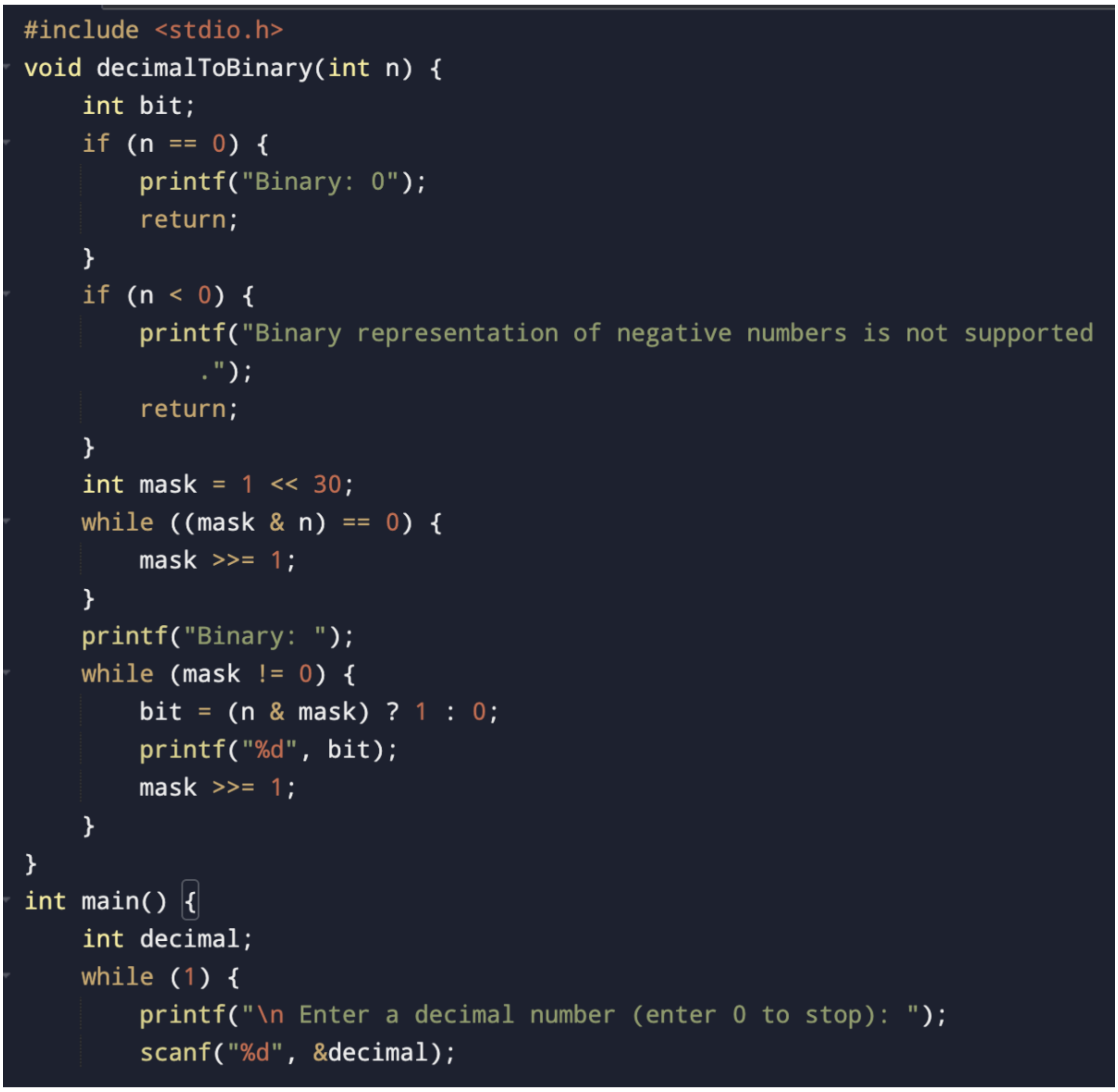


```
Enter lower and upper limits25
50
The perfect numbers are :28
```

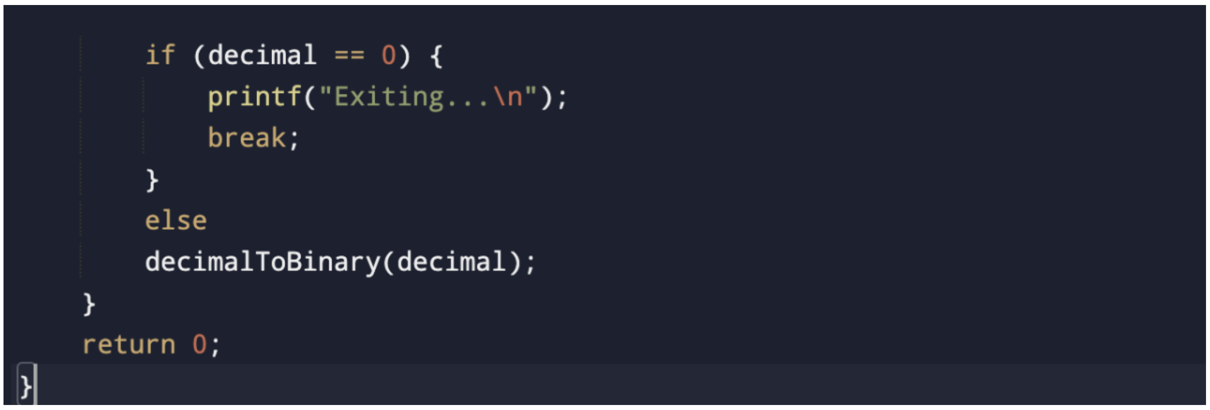
3. Write a C function for converting a given decimal number into a binary number and make use of this function to find the binary equivalent of several decimal numbers in succession until you decide to stop.



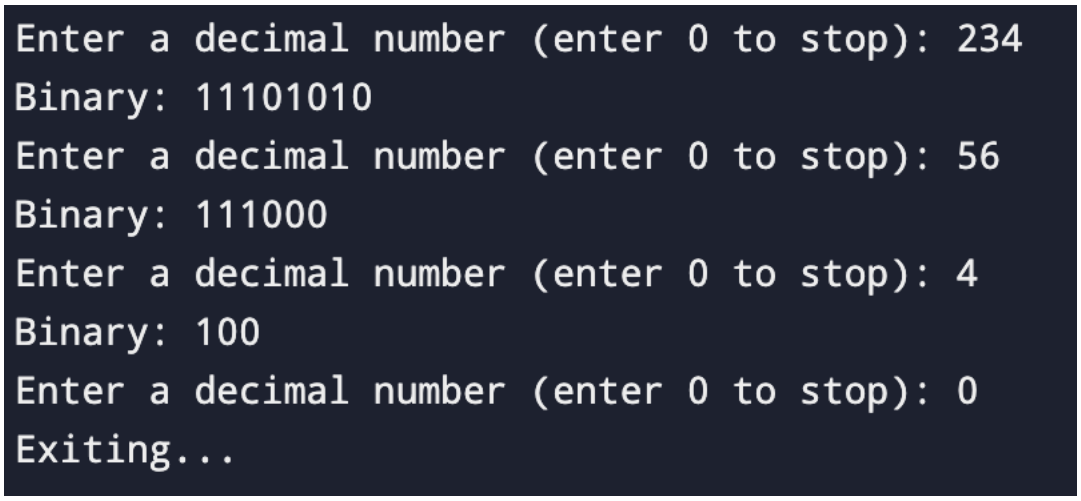
```
→ For the function decimalToBinary(int decimal)
1. Begin
2. int bit;
3. if (n==0)
        print ("Binary : 0");
        return;
   end if
4. if (n<0)
        print ("Invalid input")
        return;
   end if
5. int m = 1<<30;
6. while ((mask &n)==0)
        mask >>=1;
   end while
7. while (mask !=0)
        bit = (n& mask)? 1:0;
        print(bit);
        mask>>=1;
   end while
8. End.
```



```c
#include <stdio.h>
void decimalToBinary(int n) {
    int bit;
    if (n == 0) {
        printf("Binary: 0");
        return;
    }
    if (n < 0) {
        printf("Binary representation of negative numbers is not supported
            .");
        return;
    }
    int mask = 1 << 30;
    while ((mask & n) == 0) {
        mask >>= 1;
    }
    printf("Binary: ");
    while (mask != 0) {
        bit = (n & mask) ? 1 : 0;
        printf("%d", bit);
        mask >>= 1;
    }
}
int main() {
    int decimal;
    while (1) {
        printf("\n Enter a decimal number (enter 0 to stop): ");
        scanf("%d", &decimal);
```



```c
        if (decimal == 0) {
            printf("Exiting...\n");
            break;
        }
        else
        decimalToBinary(decimal);
    }
    return 0;
}
```



```
Enter a decimal number (enter 0 to stop): 234
Binary: 11101010
Enter a decimal number (enter 0 to stop): 56
Binary: 111000
Enter a decimal number (enter 0 to stop): 4
Binary: 100
Enter a decimal number (enter 0 to stop): 0
Exiting...
```

4.Write a C function to cyclically swap the contents of 3 variables using call by reference parameter passing method. Make use of this function to achieve cyclic swapping for the 3 variables with n cycles.

Example:

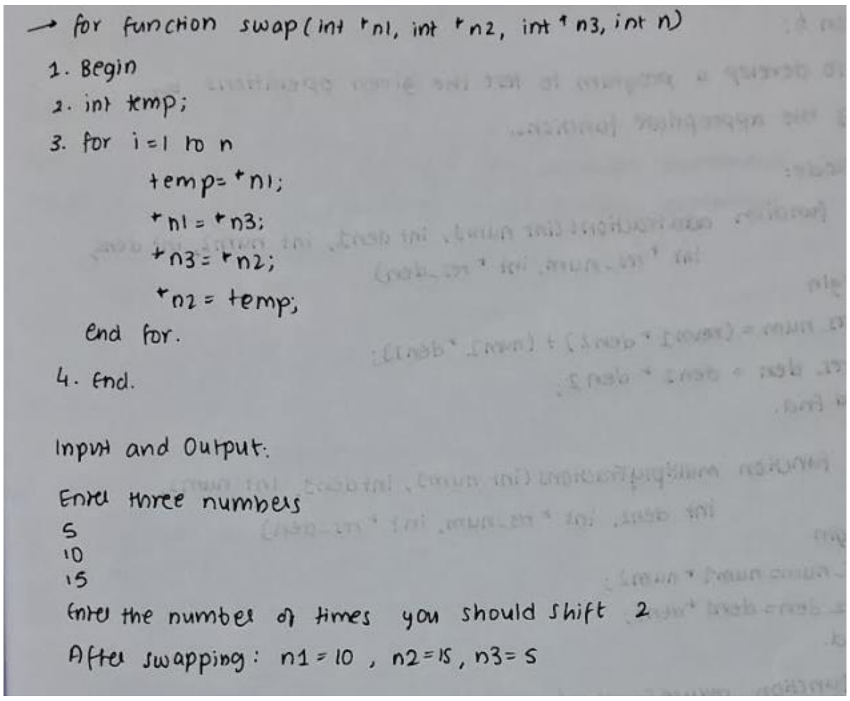
Input: a = 5 b = 10 c = 15

number of cycles: 1

output : a=15 b = 5 c = 10

number of cycles: 2

output : a=10 b = 15 c = 5

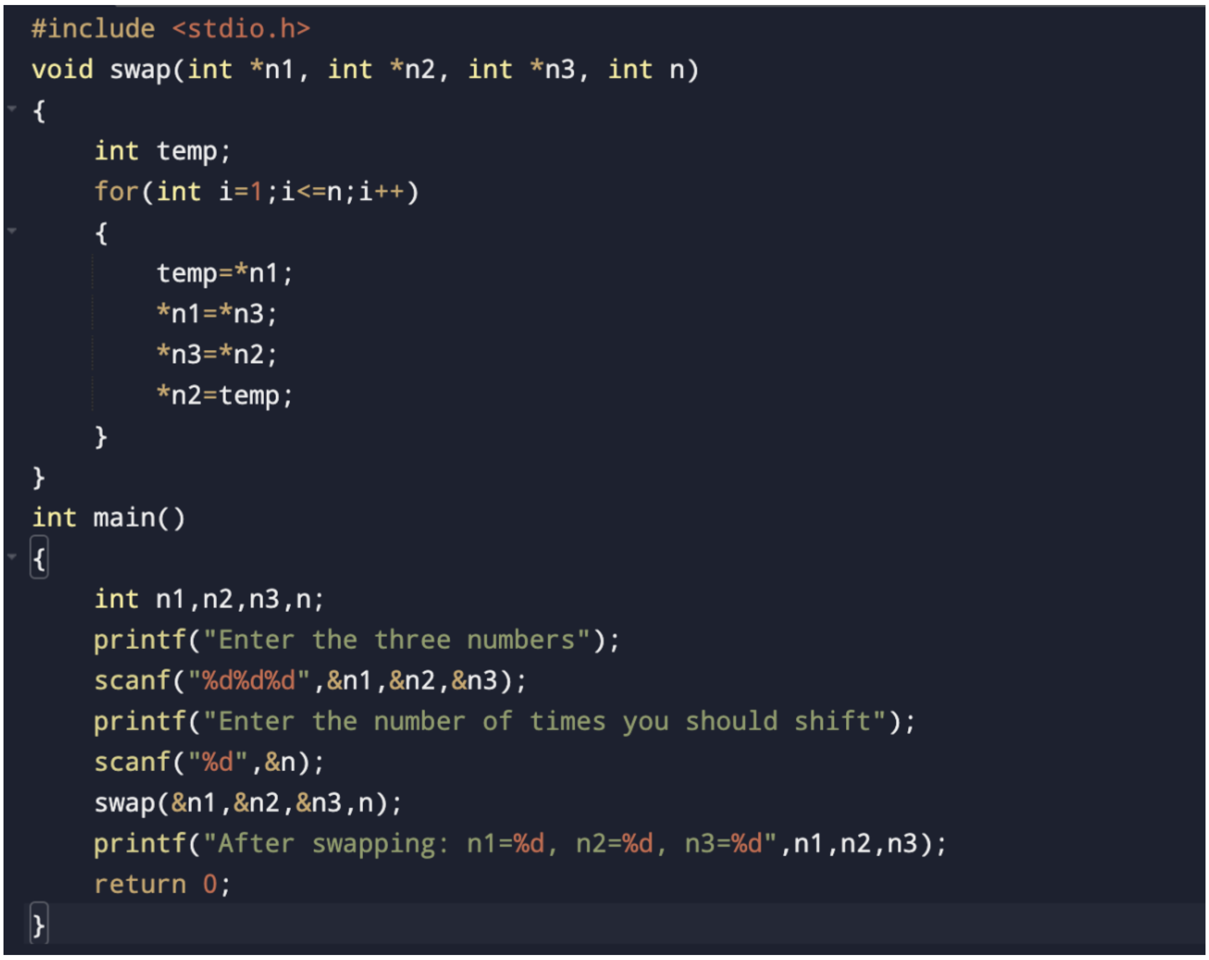


→ for function swap(int *n1, int *n2, int *n3, int n)

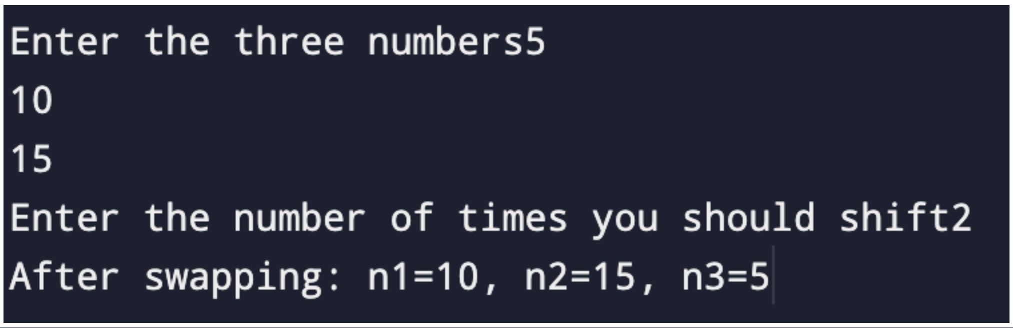
```
1. Begin
2. int temp;
3. for i=1 to n
        temp=*n1;
        *n1=*n3;
        *n3=*n2;
        *n2=temp;
   End for.
4. End.
```

Input and Output:

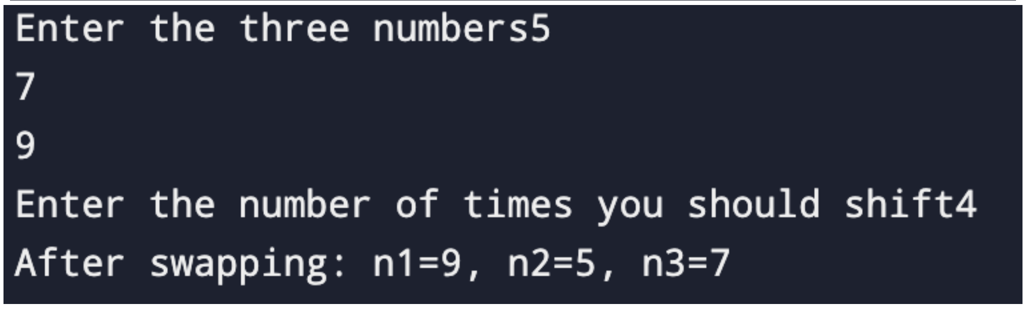
```
Enter three numbers
5
10
15
Enter the number of times you should shift  2
After swapping : n1=10 , n2=15 , n3=5
```



```c
#include <stdio.h>
void swap(int *n1, int *n2, int *n3, int n)
{
    int temp;
    for(int i=1;i<=n;i++)
    {
        temp=*n1;
        *n1=*n3;
        *n3=*n2;
        *n2=temp;
    }
}
int main()
{
    int n1,n2,n3,n;
    printf("Enter the three numbers");
    scanf("%d%d%d",&n1,&n2,&n3);
    printf("Enter the number of times you should shift");
    scanf("%d",&n);
    swap(&n1,&n2,&n3,n);
    printf("After swapping: n1=%d, n2=%d, n3=%d",n1,n2,n3);
    return 0;
}
```

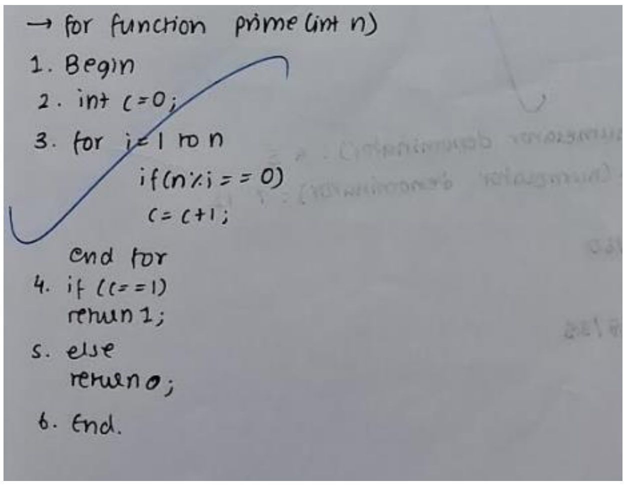


```
Enter the three numbers5
10
15
Enter the number of times you should shift2
After swapping: n1=10, n2=15, n3=5
```

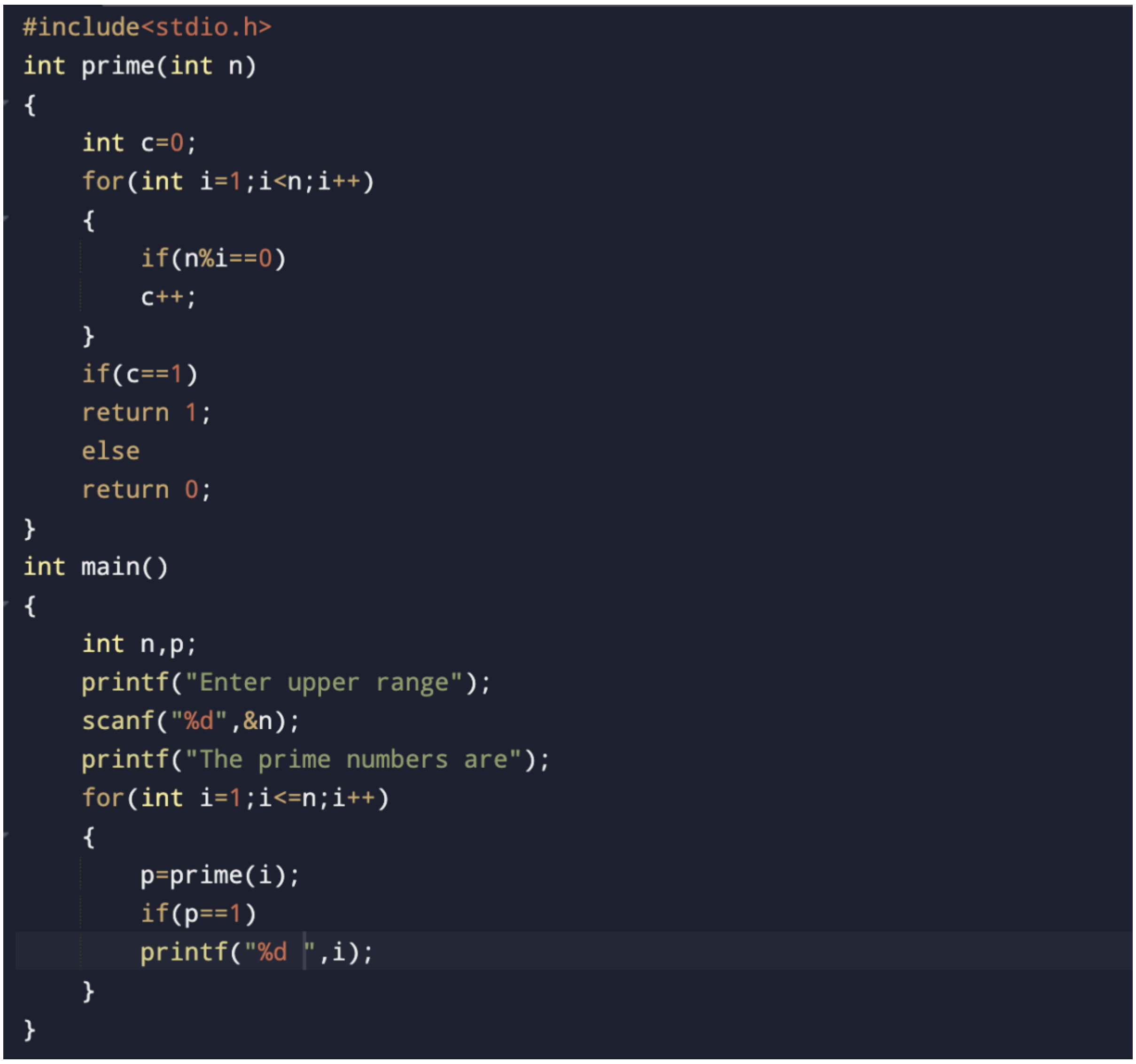


```
Enter the three numbers5
7
9
Enter the number of times you should shift4
After swapping: n1=9, n2=5, n3=7
```

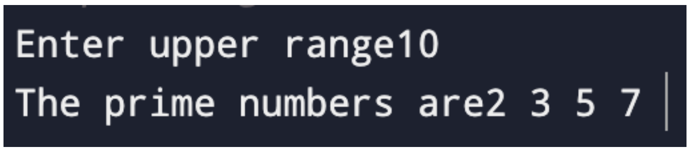
5.Write a C function to find whether a given integer is a prime number or not. Use this function to print all the prime numbers which are less than or equal to a specific input integer n.



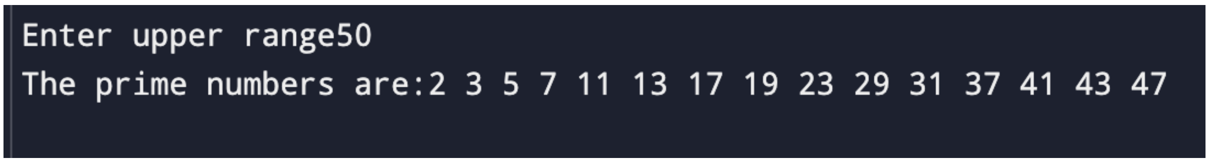
```
→ for function prime (int n)
1. Begin
2. int c=0;
3. for i=1 to n
        if(n%i==0)
        c=c+1;
   end for
4. if (c==1)
   return 1;
5. else
   return 0;
6. End.
```



```c
#include<stdio.h>
int prime(int n)
{
    int c=0;
    for(int i=1;i<n;i++)
    {
        if(n%i==0)
        c++;
    }
    if(c==1)
    return 1;
    else
    return 0;
}
int main()
{
    int n,p;
    printf("Enter upper range");
    scanf("%d",&n);
    printf("The prime numbers are");
    for(int i=1;i<=n;i++)
    {
        p=prime(i);
        if(p==1)
        printf("%d ",i);
    }
}
```



```
Enter upper range10
The prime numbers are2 3 5 7
```



```
Enter upper range50
The prime numbers are:2 3 5 7 11 13 17 19 23 29 31 37 41 43 47
```

6.Write user-defined functions to read two fractions with numerator and denominator and to perform the following operations on them and return the output also as a fraction with numerator and denominator:

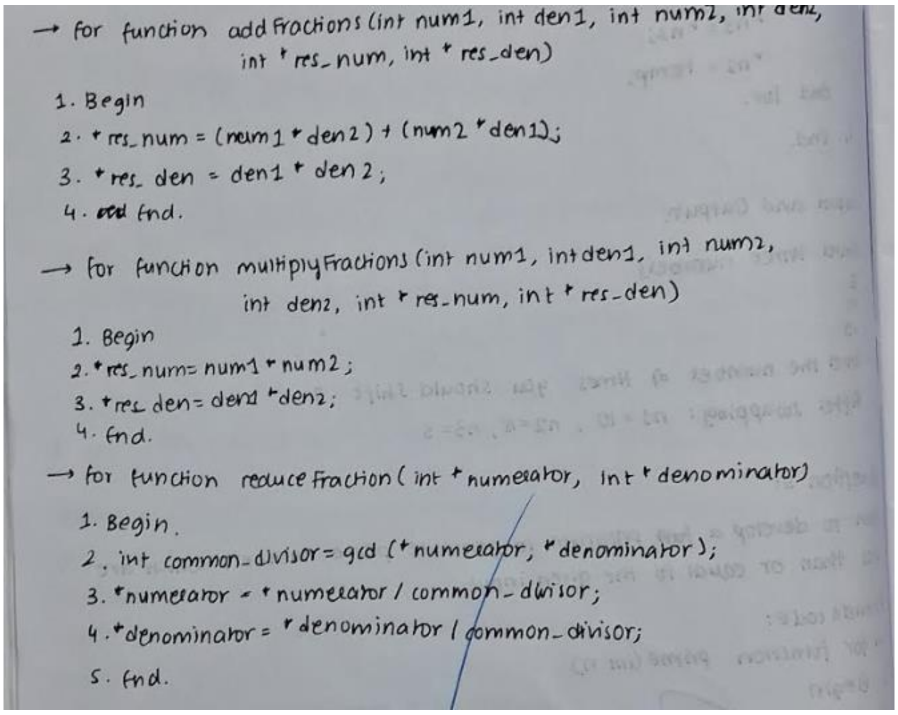
(i) Add two fractions - Input: ⅓ + ⅖ Output: 11/15

(ii) Subtract two fractions

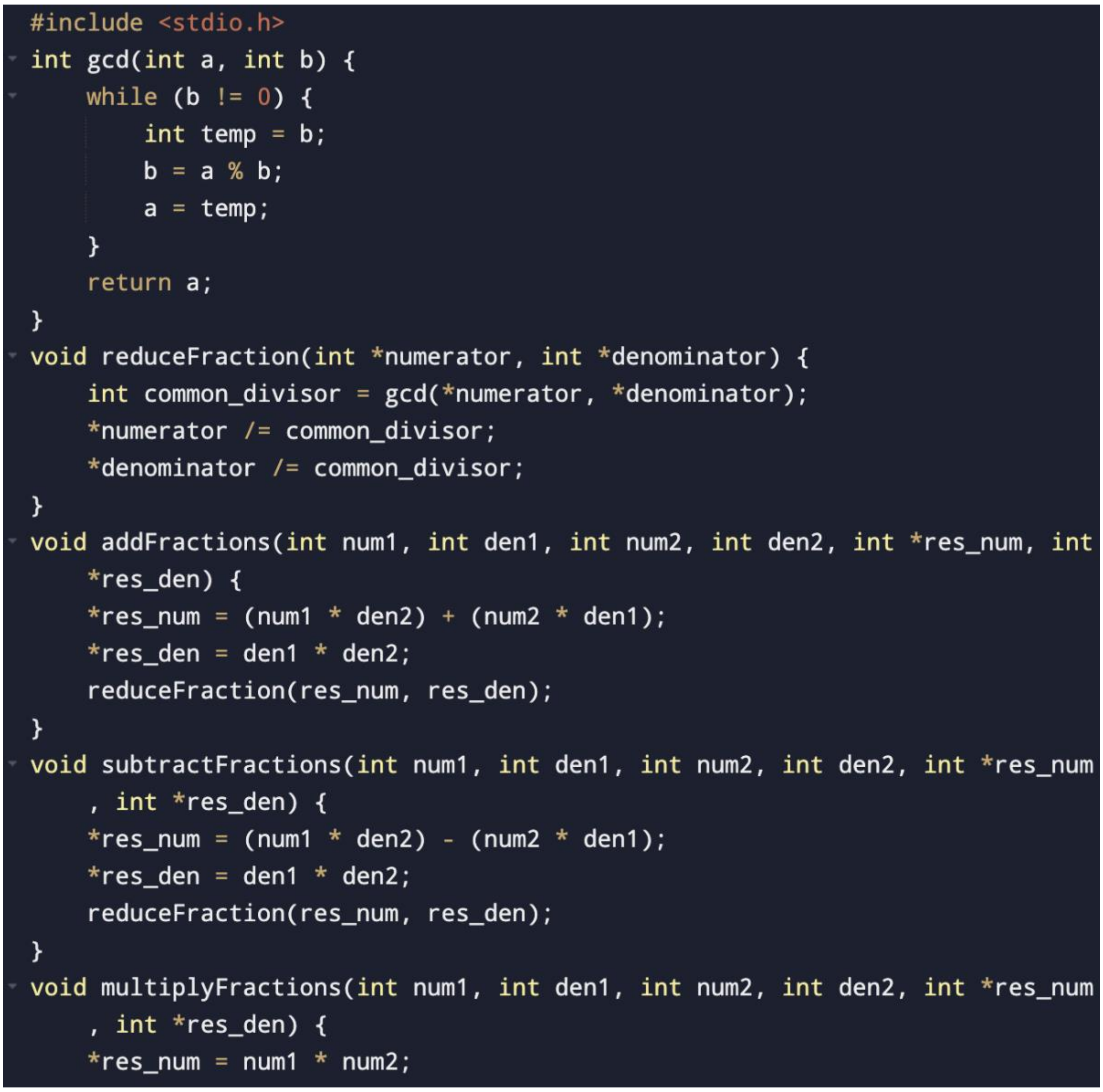
(iii) Multiply two fractions

(iv) Divide two fractions

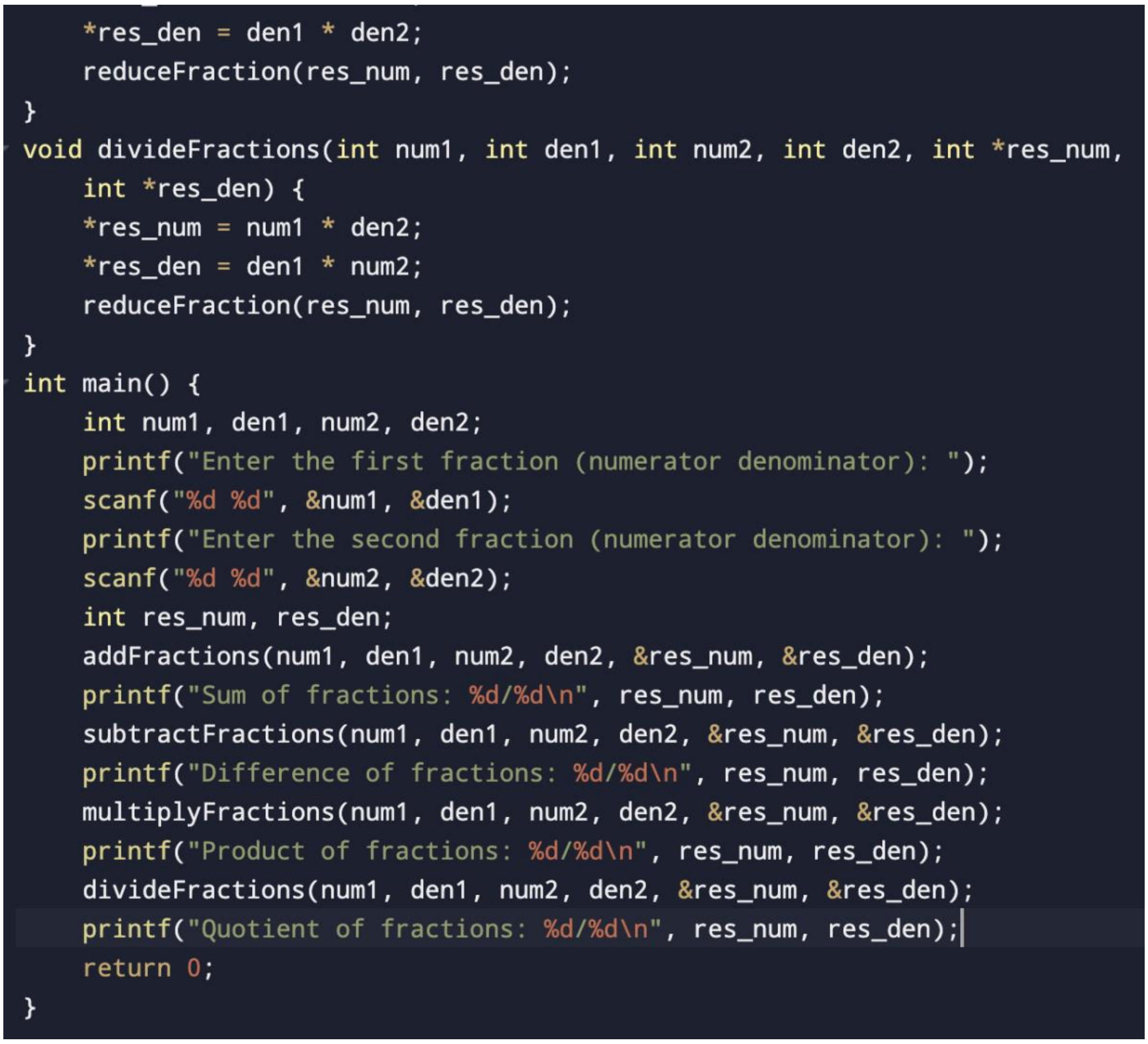
(v) Reduce a fraction



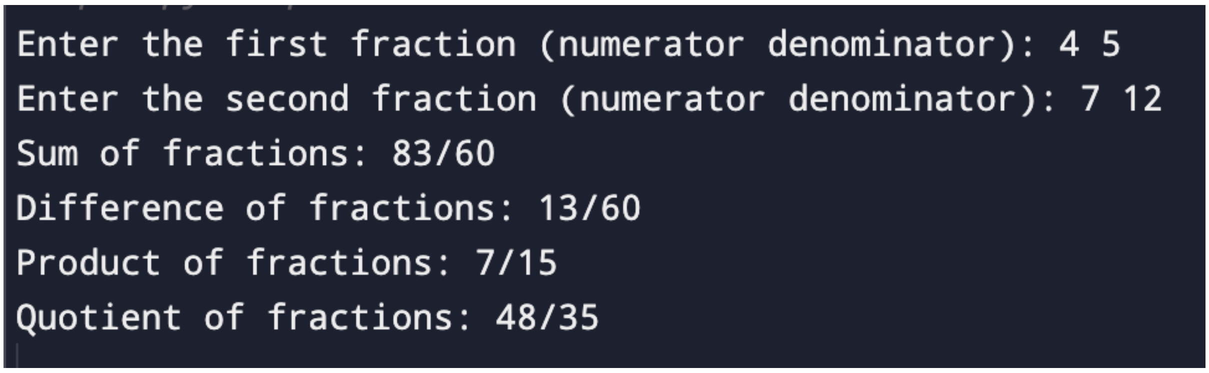
```
→ for function addFractions (int num1, int den1, int num2, int den2,
                             int *res_num, int *res_den)

  1. Begin
  2. *res_num = (num1 * den2) + (num2 * den1);
  3. *res_den = den1 * den2;
  4. End.

→ for function multiplyFractions (int num1, int den1, int num2,
                                  int den2, int *res_num, int *res_den)

  1. Begin
  2. *res_num = num1 * num2;
  3. *res_den = den1 * den2;
  4. End.

→ for function reduceFraction ( int *numerator, int *denominator)

  1. Begin.
  2. int common_divisor = gcd (*numerator, *denominator);
  3. *numerator = *numerator / common_divisor;
  4. *denominator = *denominator / common_divisor;
  5. End.
```



```c
#include <stdio.h>
int gcd(int a, int b) {
    while (b != 0) {
        int temp = b;
        b = a % b;
        a = temp;
    }
    return a;
}
void reduceFraction(int *numerator, int *denominator) {
    int common_divisor = gcd(*numerator, *denominator);
    *numerator /= common_divisor;
    *denominator /= common_divisor;
}
void addFractions(int num1, int den1, int num2, int den2, int *res_num, int
    *res_den) {
    *res_num = (num1 * den2) + (num2 * den1);
    *res_den = den1 * den2;
    reduceFraction(res_num, res_den);
}
void subtractFractions(int num1, int den1, int num2, int den2, int *res_num
    , int *res_den) {
    *res_num = (num1 * den2) - (num2 * den1);
    *res_den = den1 * den2;
    reduceFraction(res_num, res_den);
}
void multiplyFractions(int num1, int den1, int num2, int den2, int *res_num
    , int *res_den) {
    *res_num = num1 * num2;
```



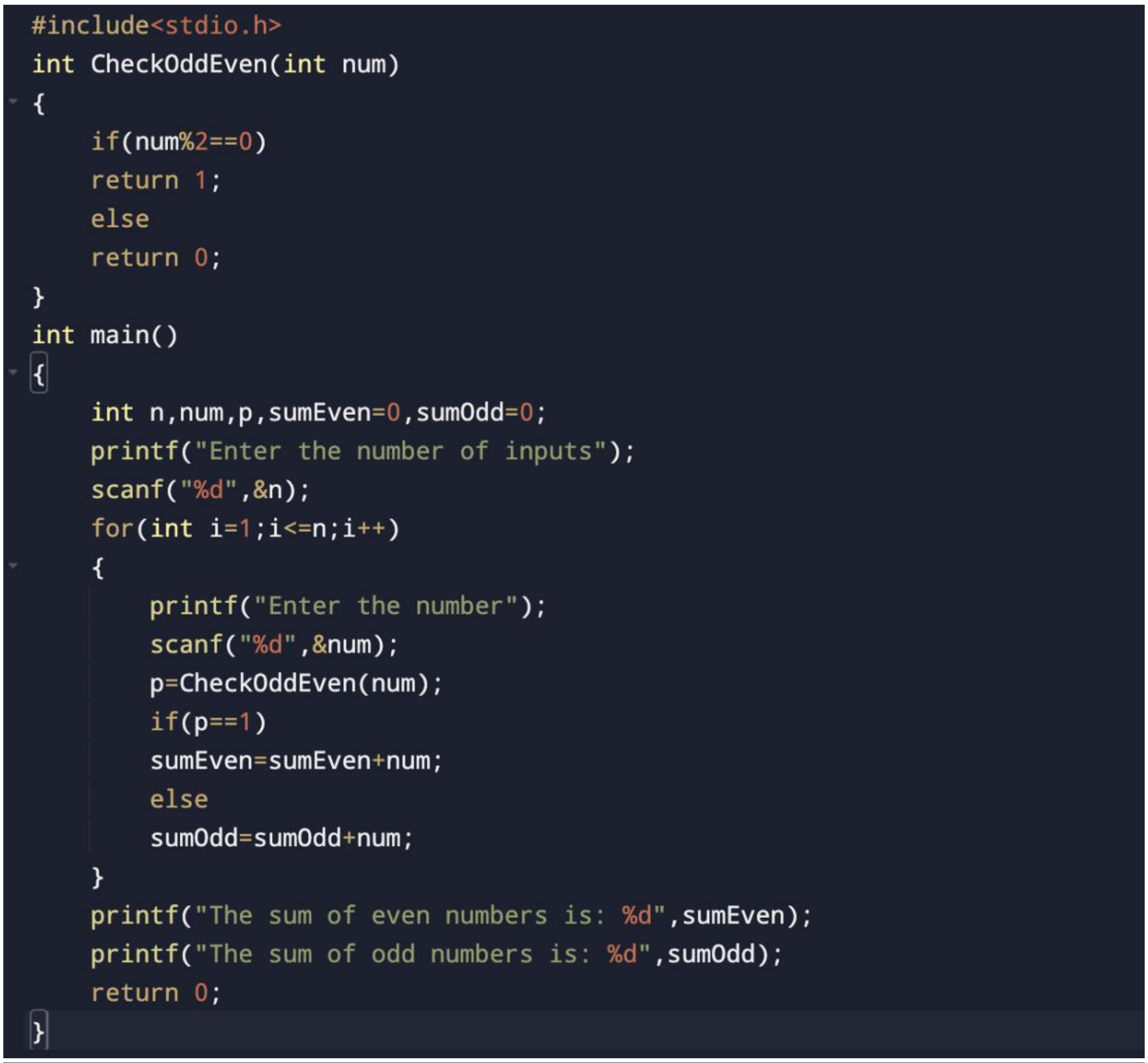
```c
    *res_den = den1 * den2;
    reduceFraction(res_num, res_den);
}
void divideFractions(int num1, int den1, int num2, int den2, int *res_num,
    int *res_den) {
    *res_num = num1 * den2;
    *res_den = den1 * num2;
    reduceFraction(res_num, res_den);
}
int main() {
    int num1, den1, num2, den2;
    printf("Enter the first fraction (numerator denominator): ");
    scanf("%d %d", &num1, &den1);
    printf("Enter the second fraction (numerator denominator): ");
    scanf("%d %d", &num2, &den2);
    int res_num, res_den;
    addFractions(num1, den1, num2, den2, &res_num, &res_den);
    printf("Sum of fractions: %d/%d\n", res_num, res_den);
    subtractFractions(num1, den1, num2, den2, &res_num, &res_den);
    printf("Difference of fractions: %d/%d\n", res_num, res_den);
    multiplyFractions(num1, den1, num2, den2, &res_num, &res_den);
    printf("Product of fractions: %d/%d\n", res_num, res_den);
    divideFractions(num1, den1, num2, den2, &res_num, &res_den);
    printf("Quotient of fractions: %d/%d\n", res_num, res_den);
    return 0;
}
```



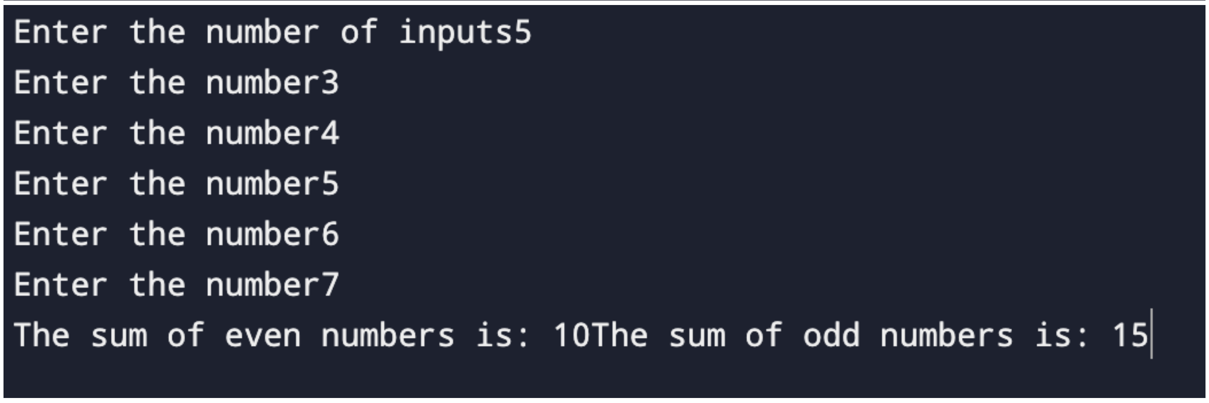
```
Enter the first fraction (numerator denominator): 4 5
Enter the second fraction (numerator denominator): 7 12
Sum of fractions: 83/60
Difference of fractions: 13/60
Product of fractions: 7/15
Quotient of fractions: 48/35
```

Additional programs for practice:

1. Write a function CheckOddEven(num) that checks if the num is odd or even; set a flag accordingly and return it. Use this function to find the sum of even and odd numbers in a given input of N.

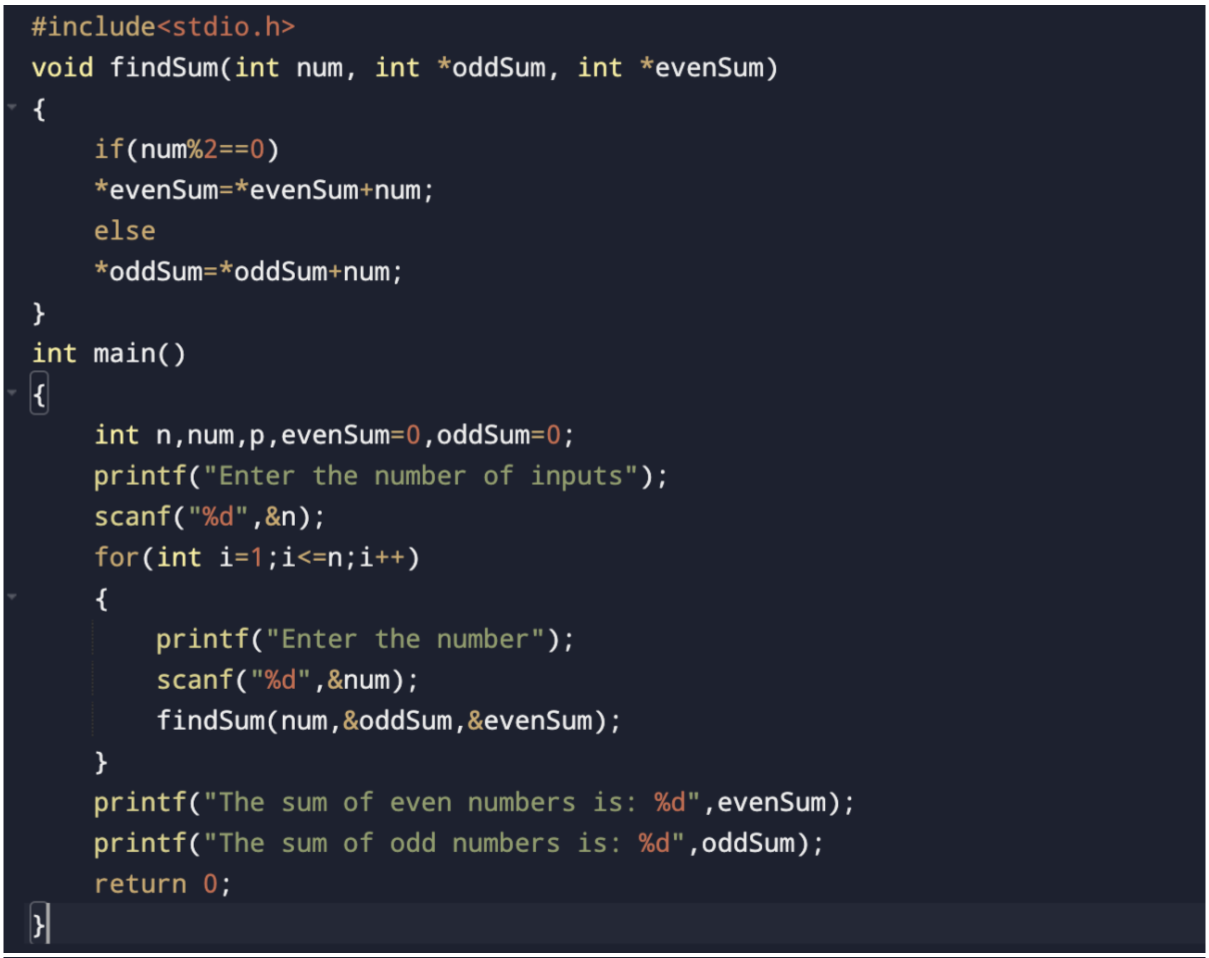


```c
#include<stdio.h>
int CheckOddEven(int num)
{
    if(num%2==0)
    return 1;
    else
    return 0;
}
int main()
{
    int n,num,p,sumEven=0,sumOdd=0;
    printf("Enter the number of inputs");
    scanf("%d",&n);
    for(int i=1;i<=n;i++)
    {
        printf("Enter the number");
        scanf("%d",&num);
        p=CheckOddEven(num);
        if(p==1)
        sumEven=sumEven+num;
        else
        sumOdd=sumOdd+num;
    }
    printf("The sum of even numbers is: %d",sumEven);
    printf("The sum of odd numbers is: %d",sumOdd);
    return 0;
}
```

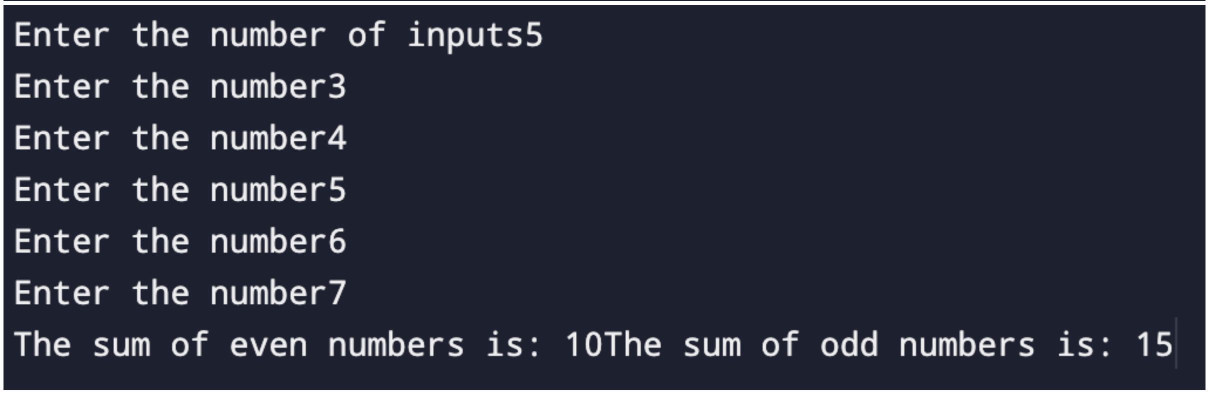


```
Enter the number of inputs5
Enter the number3
Enter the number4
Enter the number5
Enter the number6
Enter the number7
The sum of even numbers is: 10The sum of odd numbers is: 15
```

2. Add another function FindSum(num, oddSum, evenSum) to find the sum of odd and even numbers by using call by reference without returning the sums explicitly. Read the set of numbers in the main() function.

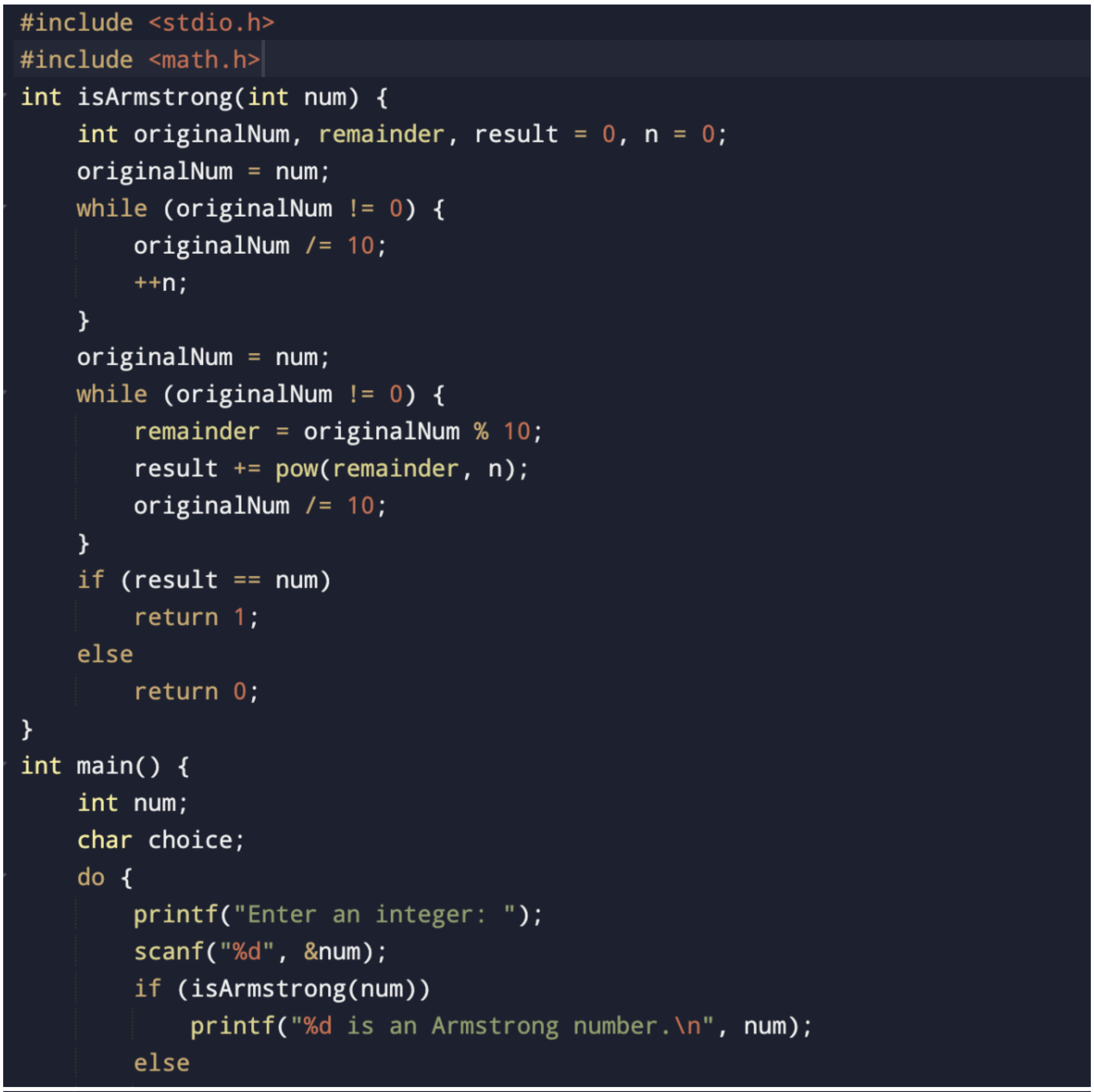


```c
#include<stdio.h>
void findSum(int num, int *oddSum, int *evenSum)
{
    if(num%2==0)
    *evenSum=*evenSum+num;
    else
    *oddSum=*oddSum+num;
}
int main()
{
    int n,num,p,evenSum=0,oddSum=0;
    printf("Enter the number of inputs");
    scanf("%d",&n);
    for(int i=1;i<=n;i++)
    {
        printf("Enter the number");
        scanf("%d",&num);
        findSum(num,&oddSum,&evenSum);
    }
    printf("The sum of even numbers is: %d",evenSum);
    printf("The sum of odd numbers is: %d",oddSum);
    return 0;
}
```

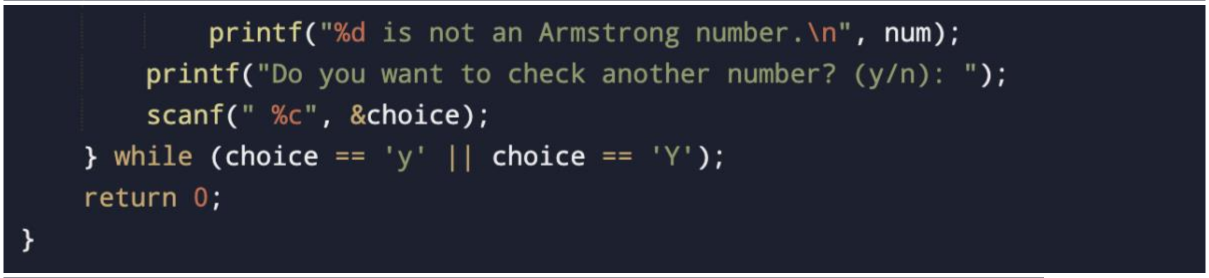


```
Enter the number of inputs5
Enter the number3
Enter the number4
Enter the number5
Enter the number6
Enter the number7
The sum of even numbers is: 10The sum of odd numbers is: 15
```

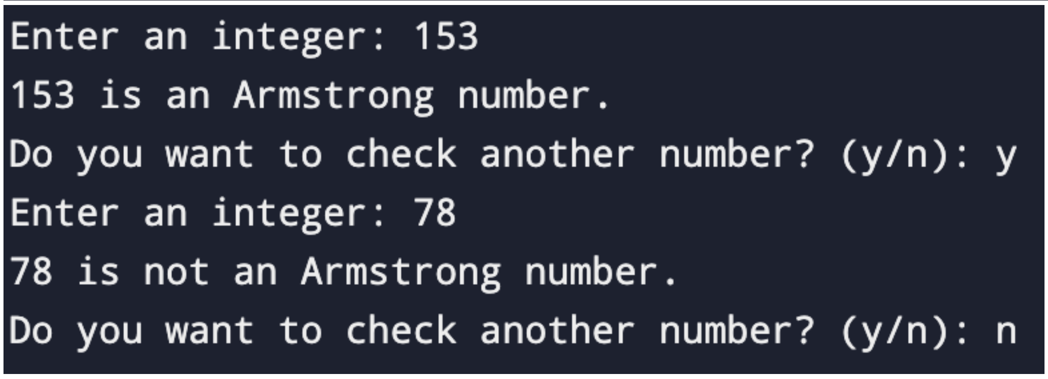
3. Write a C function to check whether a given integer is an Armstrong number or not. Use this function repeatedly from a main function for repeatedly doing this check for multiple input integers.



```c
#include <stdio.h>
#include <math.h>
int isArmstrong(int num) {
    int originalNum, remainder, result = 0, n = 0;
    originalNum = num;
    while (originalNum != 0) {
        originalNum /= 10;
        ++n;
    }
    originalNum = num;
    while (originalNum != 0) {
        remainder = originalNum % 10;
        result += pow(remainder, n);
        originalNum /= 10;
    }
    if (result == num)
        return 1;
    else
        return 0;
}
int main() {
    int num;
    char choice;
    do {
        printf("Enter an integer: ");
        scanf("%d", &num);
        if (isArmstrong(num))
            printf("%d is an Armstrong number.\n", num);
        else
```



```c
            printf("%d is not an Armstrong number.\n", num);
        printf("Do you want to check another number? (y/n): ");
        scanf(" %c", &choice);
    } while (choice == 'y' || choice == 'Y');
    return 0;
}
```



```
Enter an integer: 153
153 is an Armstrong number.
Do you want to check another number? (y/n): y
Enter an integer: 78
78 is not an Armstrong number.
Do you want to check another number? (y/n): n
```