UCS 2201 Fundamentals and Practice of Software Development

A2: Practicing Looping Constructs of C

Batch 2023-2027
Name: SK Shivaanee
Register ID: 2310257
Section: CSE B

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

Learning Outcome:

You will be able to familiarize the following basic features of C

● Looping constructs (for, while, do-while)

Best Practices:

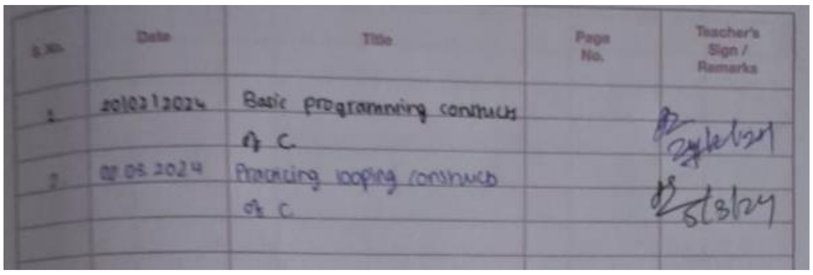
• Proper variable naming conventions should be followed.

• Proper Indentation should be used

• Appropriate read and print statements should be used.

• Appropriate comments should be used.

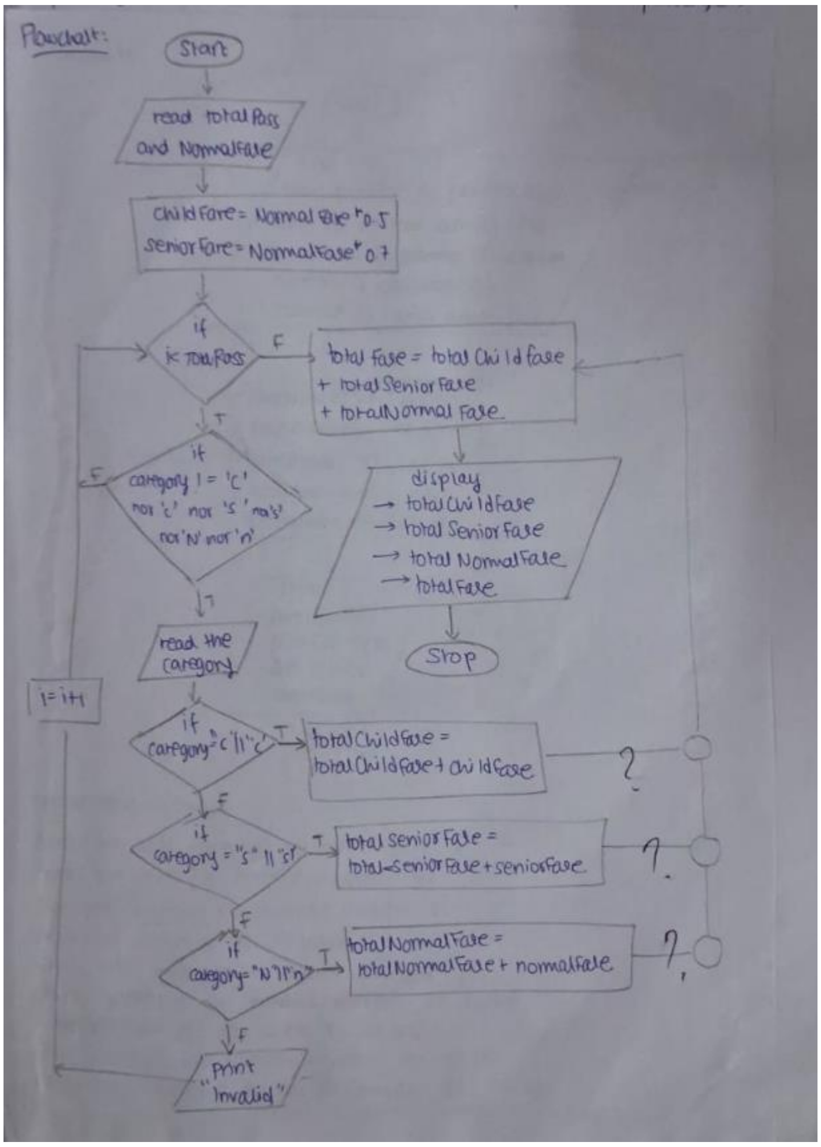
Assignment: Solve the following problems by implementing in C. (CO7, K3, 1.3.1, 1.4.1, 2.1.2, 2.1.3, 4.2.1, 14.2.1, 14.2.2)



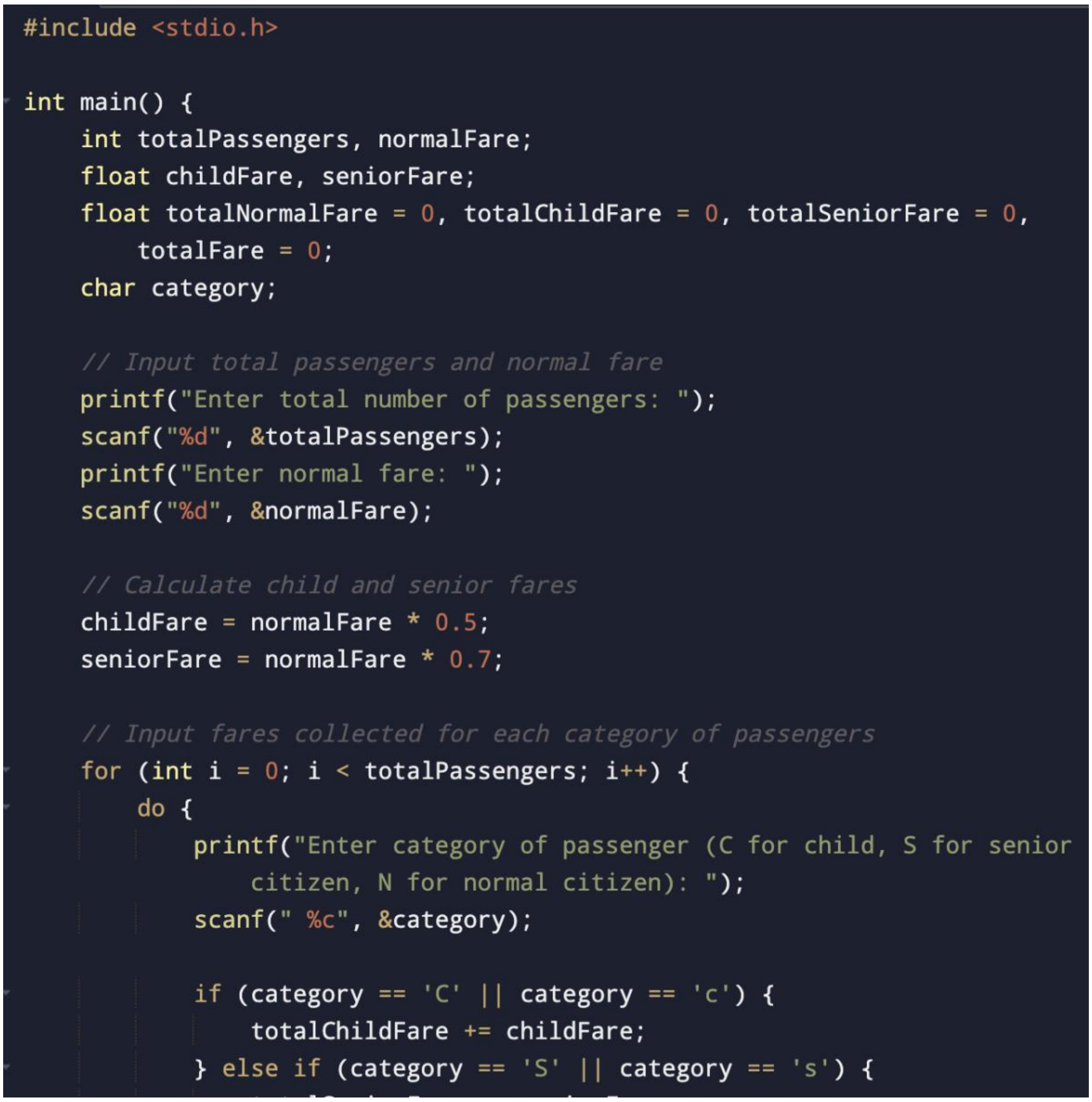
S.No.	Date	Title	Page No.	Teacher's Sign / Remarks
1	20/02/2024	Basic programming constructs of C		[illegible] 29/2/24
2	02.03.2024	Practicing looping constructs of C		[illegible] 5/3/24

1.In a bus, a total of X passengers are traveling. Passengers can be children aged less than 12 years, senior citizens (above 65), or normal citizens. The ticket fare is categorized as follows:

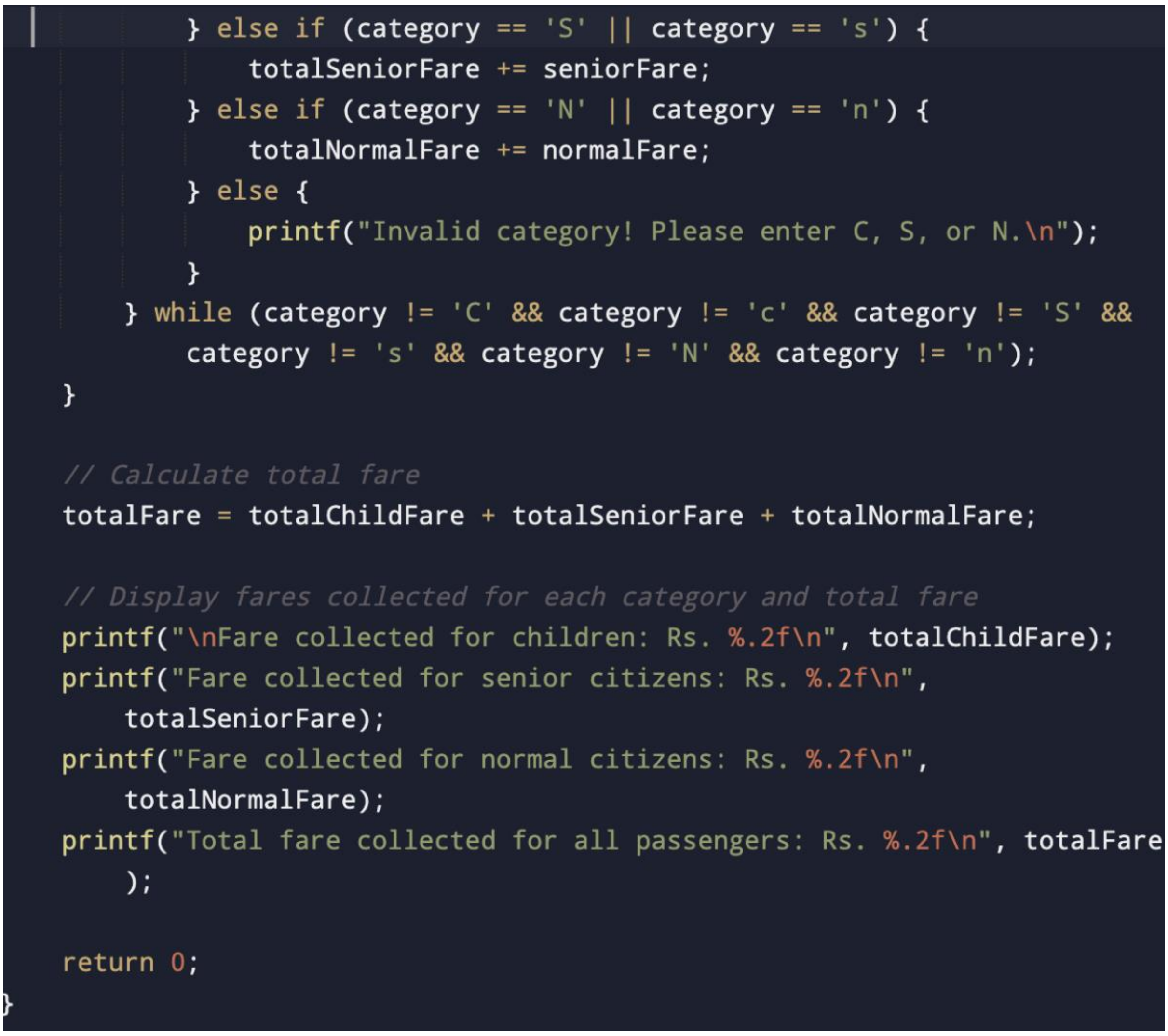
For normal citizens, it is Rs. P1.
For children, it is 50% less than the fare of normal citizens.
For senior citizens, it is 30% less than the fare of normal citizens.
Calculate the bus fare collected for each category of passengers. Also calculate the total fare for X passengers. a. Identify the inputs required to solve the problem. b. Devise a solution and represent the same using a flowchart. c. Develop a program to calculate the bus fare collected for each category of passengers. Also, calculate the total fare for X passengers.



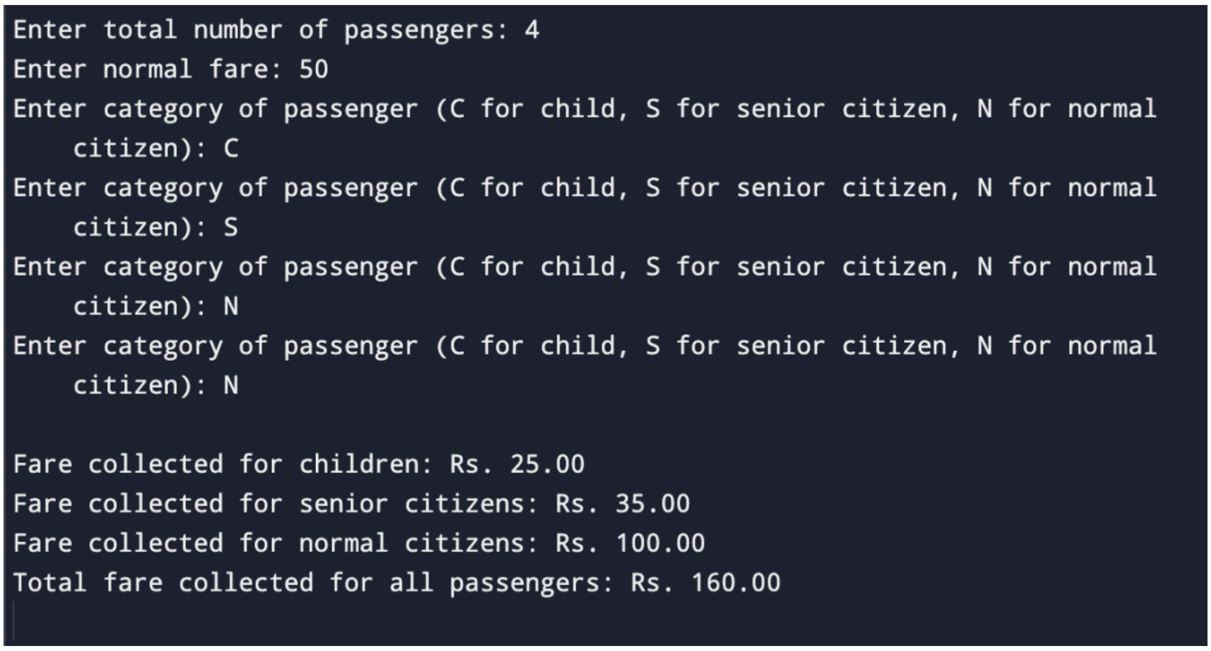
Flowchart:
Start
read totalPass and NormalFare
ChildFare = NormalFare * 0.5
seniorFare = NormalFare * 0.7
if i< TotalPass
F
T
total Fare = total Childfare + totalSeniorFare + totalNormalFare.
display
→ totalChildFare
→ total SeniorFare
→ total NormalFare
→ totalFare
Stop
if category != 'C' nor 'c' nor 'S' nor 's' nor 'N' nor 'n'
F
T
read the category
i=i+1
if category="C"||"c"
T
totalChildFare = totalChildFare + childFare
?
F
if category = "S" || "s"
T
total seniorFare = total seniorFare + seniorFare.
?
F
if category="N"||"n"
T
totalNormalFare = totalNormalFare + normalFare
?
F
Print "Invalid"



```c
#include <stdio.h>

int main() {
    int totalPassengers, normalFare;
    float childFare, seniorFare;
    float totalNormalFare = 0, totalChildFare = 0, totalSeniorFare = 0,
        totalFare = 0;
    char category;

    // Input total passengers and normal fare
    printf("Enter total number of passengers: ");
    scanf("%d", &totalPassengers);
    printf("Enter normal fare: ");
    scanf("%d", &normalFare);

    // Calculate child and senior fares
    childFare = normalFare * 0.5;
    seniorFare = normalFare * 0.7;

    // Input fares collected for each category of passengers
    for (int i = 0; i < totalPassengers; i++) {
        do {
            printf("Enter category of passenger (C for child, S for senior
                citizen, N for normal citizen): ");
            scanf(" %c", &category);

            if (category == 'C' || category == 'c') {
                totalChildFare += childFare;
            } else if (category == 'S' || category == 's') {
```



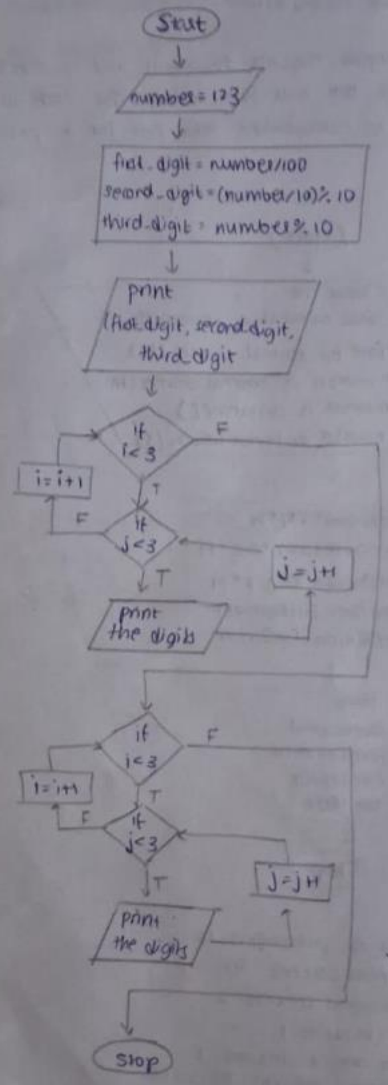
```c
            } else if (category == 'S' || category == 's') {
                totalSeniorFare += seniorFare;
            } else if (category == 'N' || category == 'n') {
                totalNormalFare += normalFare;
            } else {
                printf("Invalid category! Please enter C, S, or N.\n");
            }
        } while (category != 'C' && category != 'c' && category != 'S' &&
            category != 's' && category != 'N' && category != 'n');
    }

    // Calculate total fare
    totalFare = totalChildFare + totalSeniorFare + totalNormalFare;

    // Display fares collected for each category and total fare
    printf("\nFare collected for children: Rs. %.2f\n", totalChildFare);
    printf("Fare collected for senior citizens: Rs. %.2f\n",
        totalSeniorFare);
    printf("Fare collected for normal citizens: Rs. %.2f\n",
        totalNormalFare);
    printf("Total fare collected for all passengers: Rs. %.2f\n", totalFare
        );

    return 0;
}
```

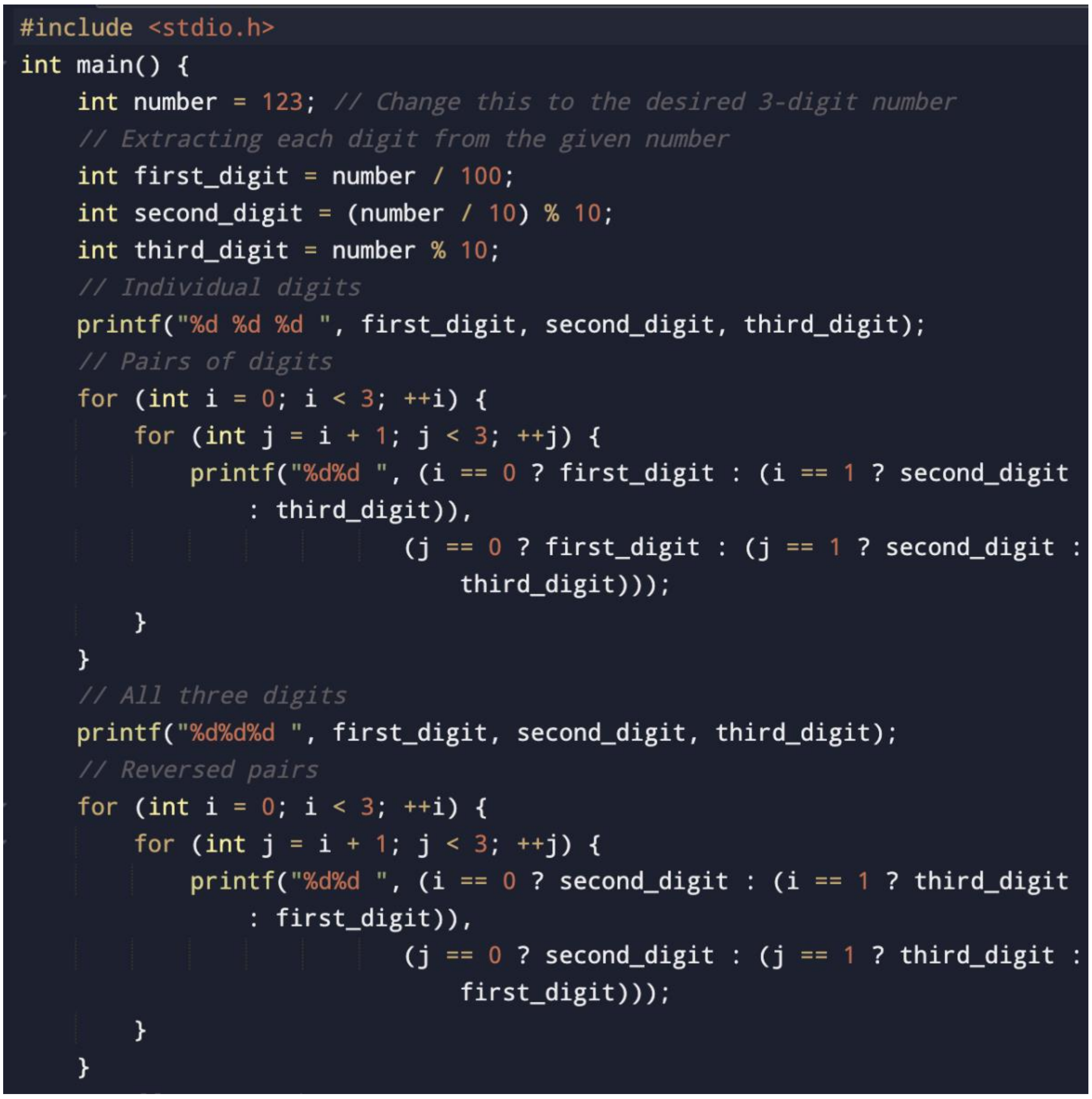


```
Enter total number of passengers: 4
Enter normal fare: 50
Enter category of passenger (C for child, S for senior citizen, N for normal
    citizen): C
Enter category of passenger (C for child, S for senior citizen, N for normal
    citizen): S
Enter category of passenger (C for child, S for senior citizen, N for normal
    citizen): N
Enter category of passenger (C for child, S for senior citizen, N for normal
    citizen): N

Fare collected for children: Rs. 25.00
Fare collected for senior citizens: Rs. 35.00
Fare collected for normal citizens: Rs. 100.00
Total fare collected for all passengers: Rs. 160.00
```

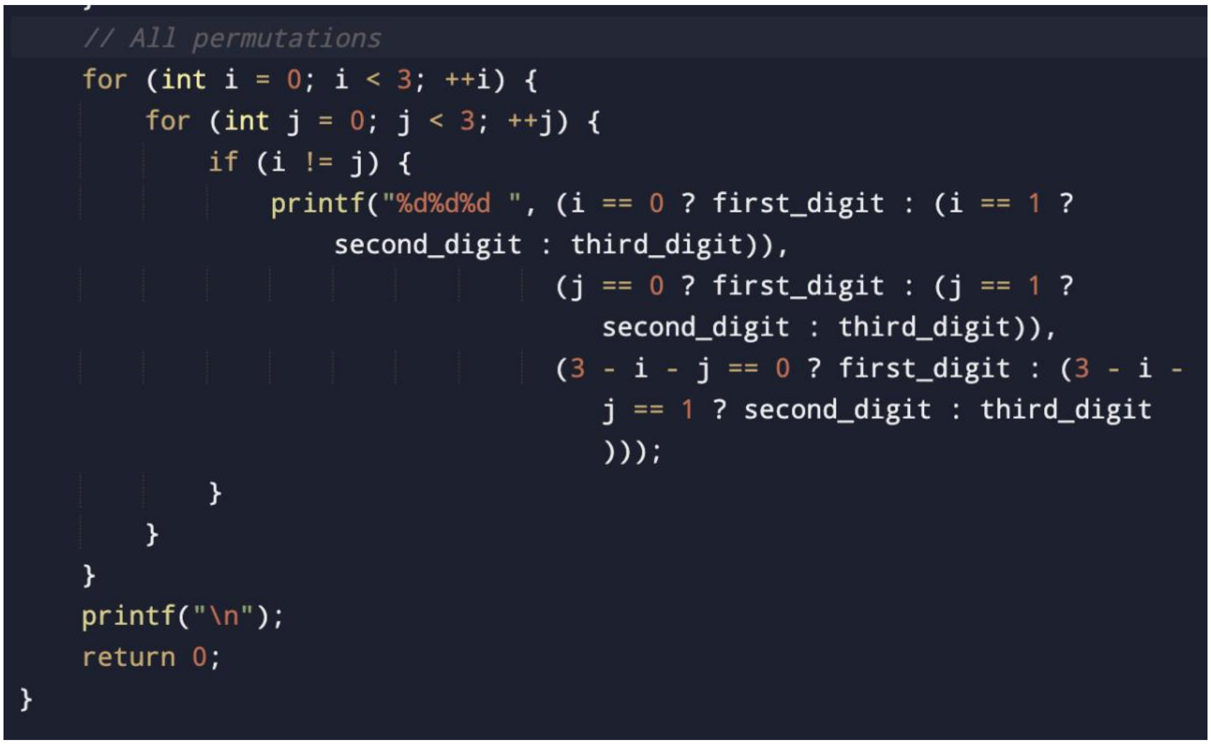
2.Generate all possible combinations of a given 3 digit number. (Eg:- 123 -> {1,2,3, 12, 13, 21, 23, 31, 32, 123, 132, 213, 231, 312, 321})

Flowchart:

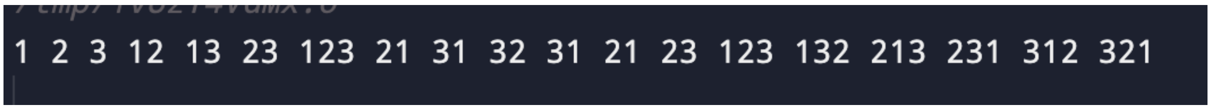

Start
number = 123
first_digit = number/100
second_digit = (number/10)% 10
third_digit = number% 10
print
(first_digit, second_digit,
third_digit
if
i < 3
F
T
i = i+1
F
if
j < 3
T
j = j+1
print
the digits
if
i<3
F
T
i = i+1
F
if
j<3
T
j = j+1
print
the digits
Stop



```c
#include <stdio.h>
int main() {
    int number = 123; // Change this to the desired 3-digit number
    // Extracting each digit from the given number
    int first_digit = number / 100;
    int second_digit = (number / 10) % 10;
    int third_digit = number % 10;
    // Individual digits
    printf("%d %d %d ", first_digit, second_digit, third_digit);
    // Pairs of digits
    for (int i = 0; i < 3; ++i) {
        for (int j = i + 1; j < 3; ++j) {
            printf("%d%d ", (i == 0 ? first_digit : (i == 1 ? second_digit
                : third_digit)),
                           (j == 0 ? first_digit : (j == 1 ? second_digit :
                               third_digit)));
        }
    }
    // All three digits
    printf("%d%d%d ", first_digit, second_digit, third_digit);
    // Reversed pairs
    for (int i = 0; i < 3; ++i) {
        for (int j = i + 1; j < 3; ++j) {
            printf("%d%d ", (i == 0 ? second_digit : (i == 1 ? third_digit
                : first_digit)),
                           (j == 0 ? second_digit : (j == 1 ? third_digit :
                               first_digit)));
        }
    }
```

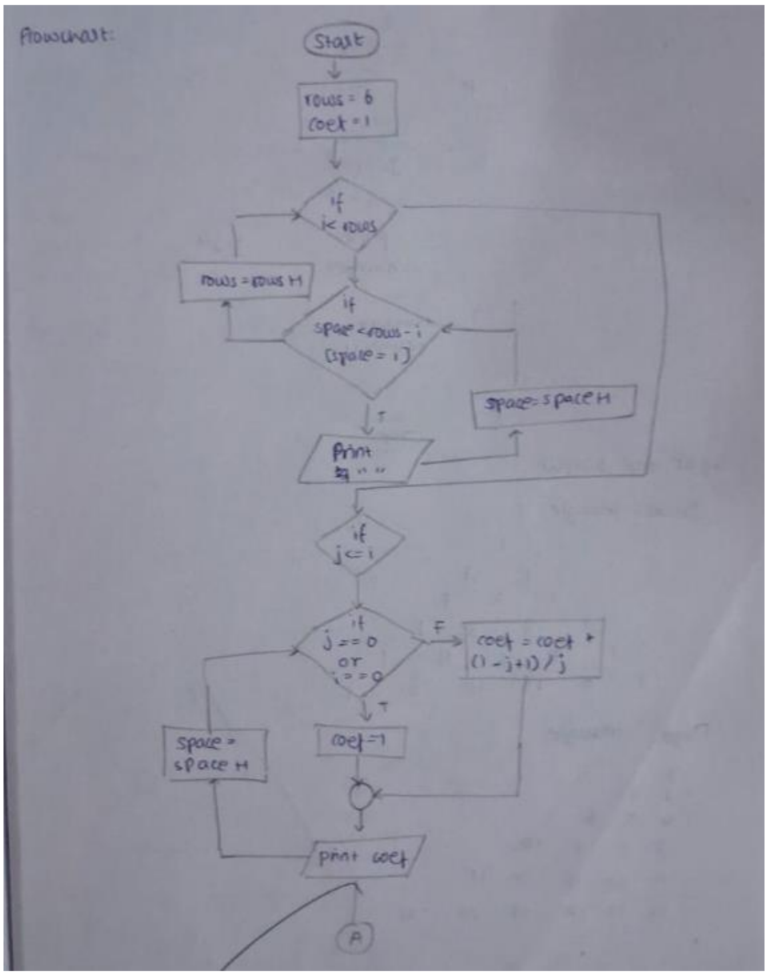


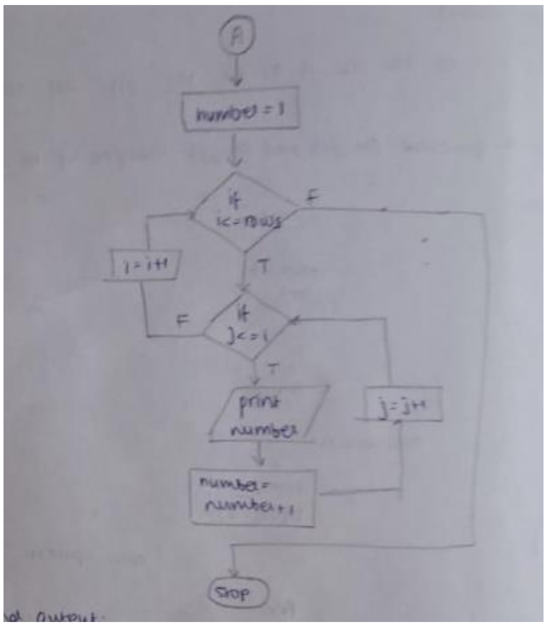
```c
    // All permutations
    for (int i = 0; i < 3; ++i) {
        for (int j = 0; j < 3; ++j) {
            if (i != j) {
                printf("%d%d%d ", (i == 0 ? first_digit : (i == 1 ?
                    second_digit : third_digit)),
                                 (j == 0 ? first_digit : (j == 1 ?
                                    second_digit : third_digit)),
                                 (3 - i - j == 0 ? first_digit : (3 - i -
                                    j == 1 ? second_digit : third_digit
                                    )));
            }
        }
    }
    printf("\n");
    return 0;
}
```

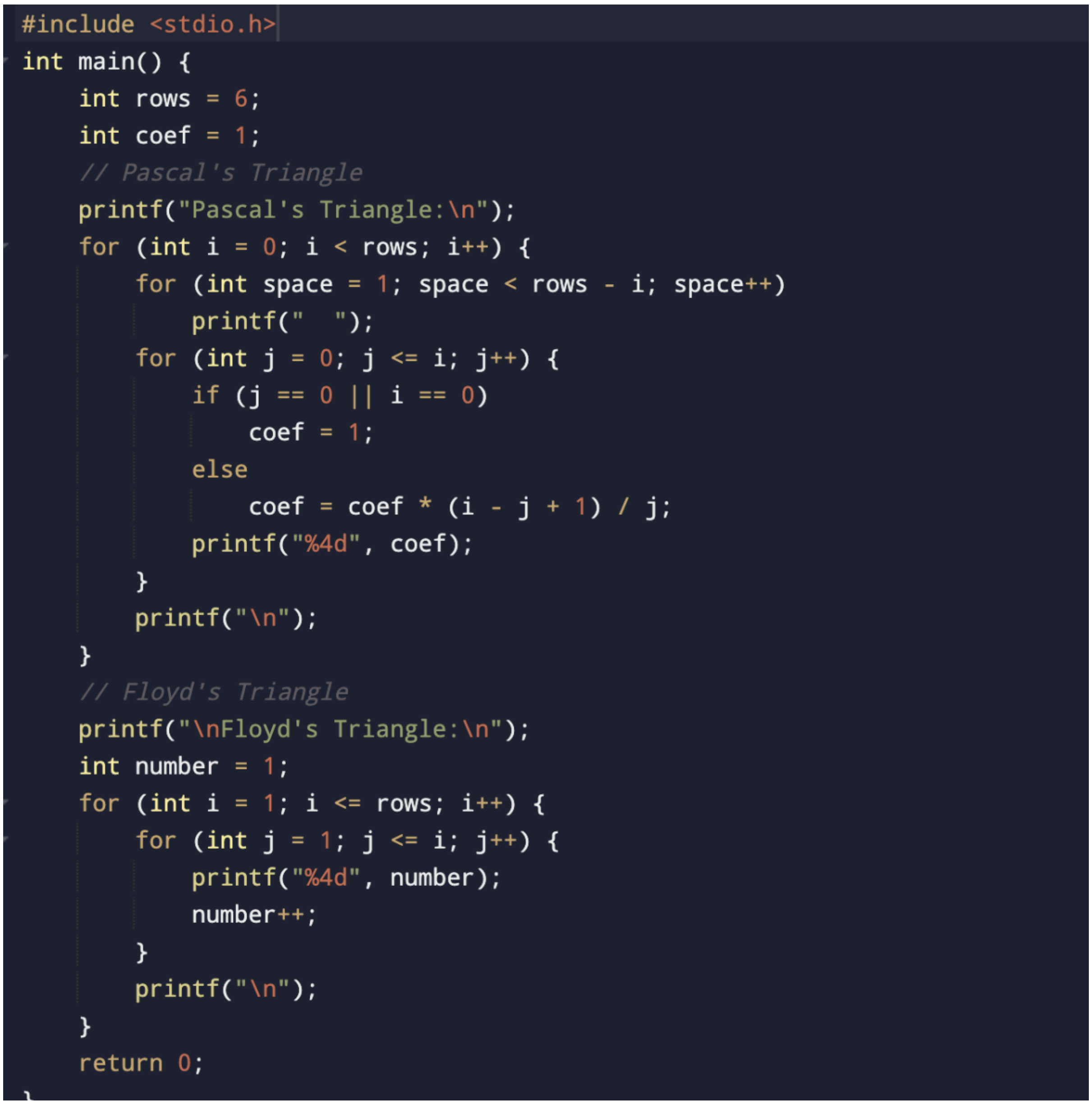


```
1 2 3 12 13 23 123 21 31 32 31 21 23 123 132 213 231 312 321
```

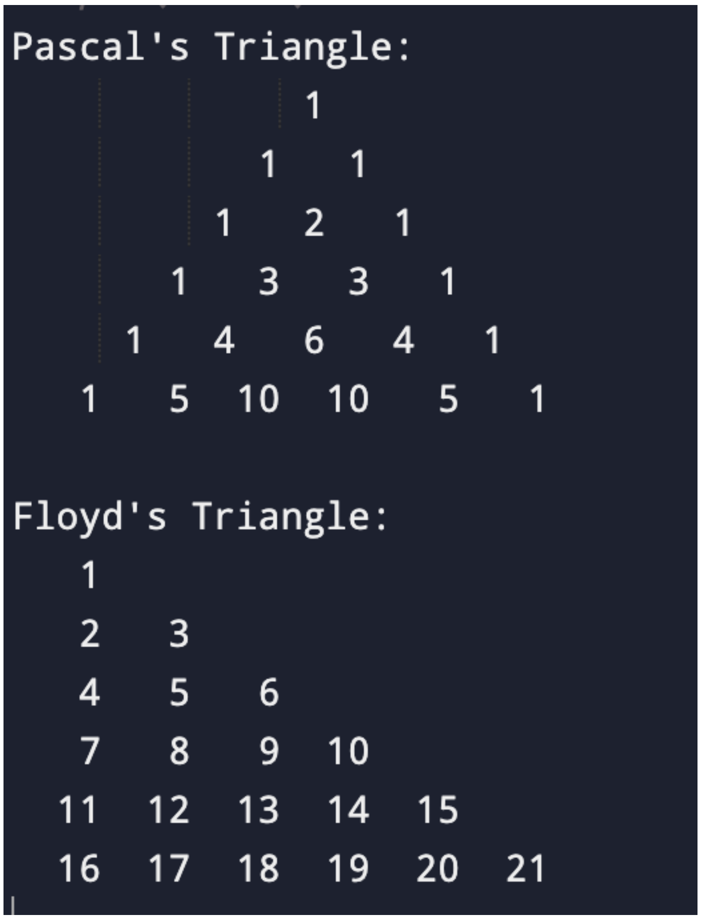
3. Generate Pascal’s and Floyd’s triangles up to 6 levels.


Flowchart:
Start
rows = 6
coef = 1
if
i< rows
rows = rows +1
if
space < rows - i
(space = 1)
T
space = space +1
Print
" "
if
j<= i
if
j == 0
or
i == 0
F
coef = coef *
(i - j+1) / j
T
space =
space +1
coef = 1
print coef
A


A
number = 1
if
i<= rows
F
i = i+1
T
F
if
j<= i
T
print
number
j = j+1
number =
number + 1
stop



```c
#include <stdio.h>
int main() {
    int rows = 6;
    int coef = 1;
    // Pascal's Triangle
    printf("Pascal's Triangle:\n");
    for (int i = 0; i < rows; i++) {
        for (int space = 1; space < rows - i; space++)
            printf("  ");
        for (int j = 0; j <= i; j++) {
            if (j == 0 || i == 0)
                coef = 1;
            else
                coef = coef * (i - j + 1) / j;
            printf("%4d", coef);
        }
        printf("\n");
    }
    // Floyd's Triangle
    printf("\nFloyd's Triangle:\n");
    int number = 1;
    for (int i = 1; i <= rows; i++) {
        for (int j = 1; j <= i; j++) {
            printf("%4d", number);
            number++;
        }
        printf("\n");
    }
    return 0;
```



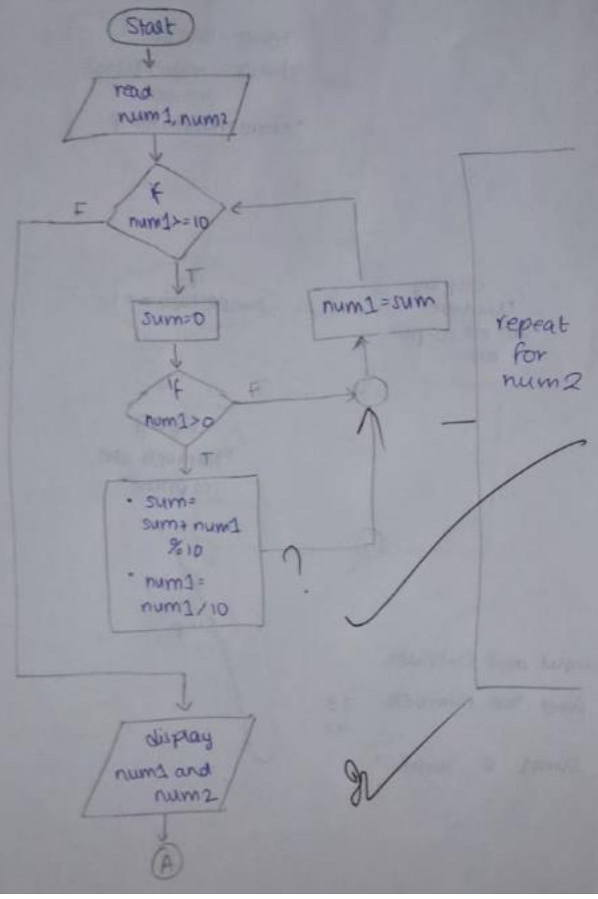
```
Pascal's Triangle:
                 1
              1     1
           1     2     1
        1     3     3     1
     1     4     6     4     1
  1     5    10    10     5     1

Floyd's Triangle:
  1
  2     3
  4     5     6
  7     8     9    10
 11    12    13    14    15
 16    17    18    19    20    21
```

4. Read 2 numbers with any number of digits.Find the sum of digits for each number until the number of digits becomes one. Check whether the resultant numbers are co-prime or not. Co-prime numbers mean pairs of numbers that do not have any common factor other than 1. Repeat checking for many pairs of numbers until you wish to check.

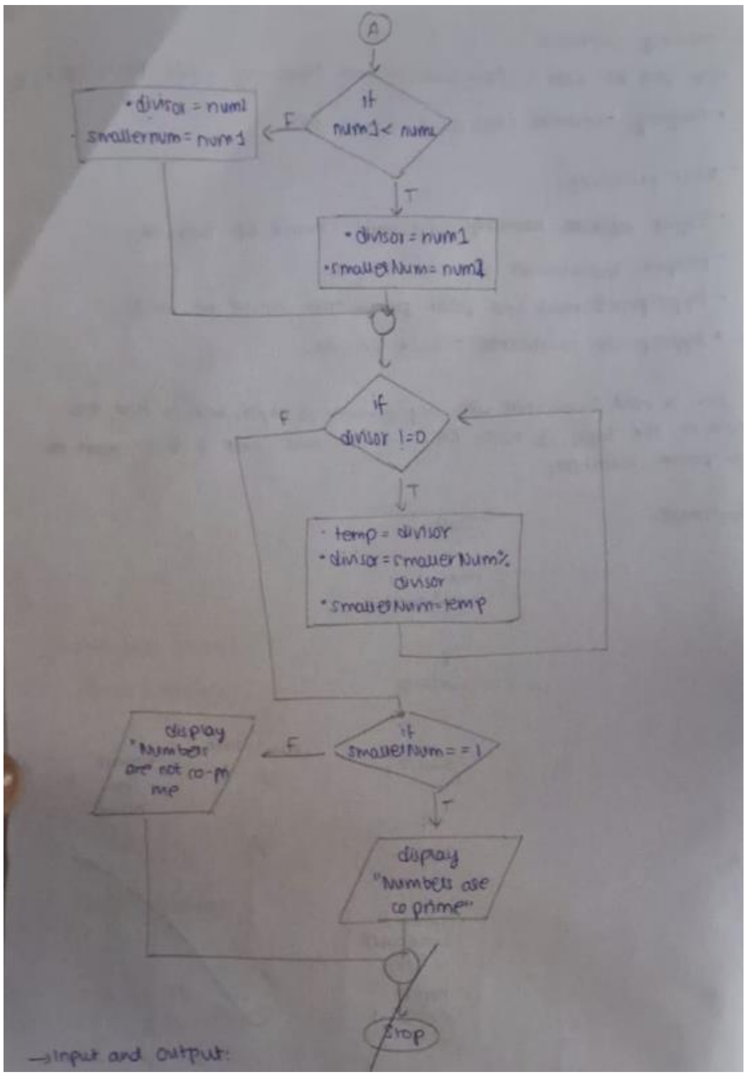
E.g.Nol: 345 No2: 2378

Sum1:12->3 Sum2: 20->2

Flowchart:

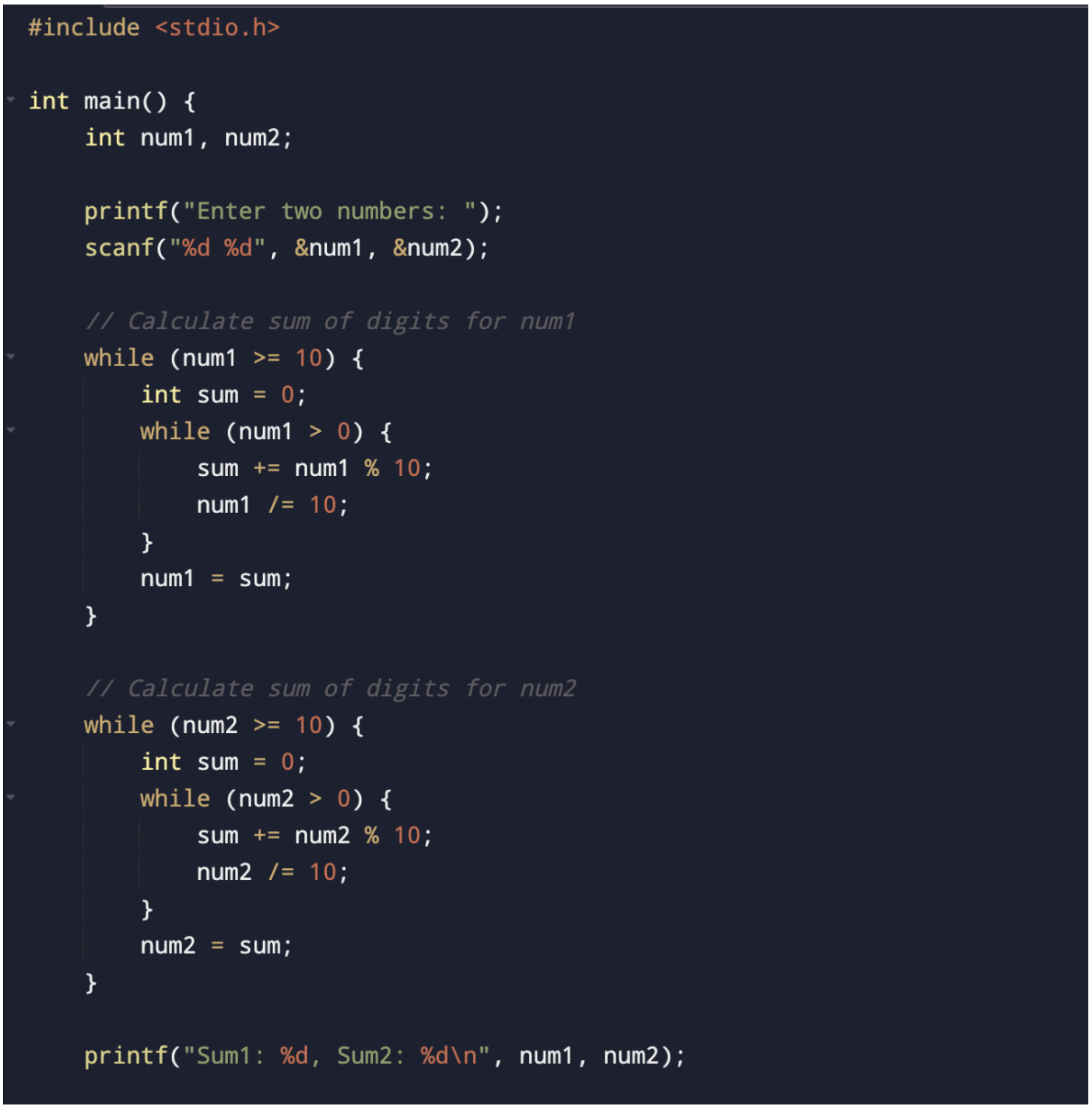


Start
read num1, num2
if num1>=10
F
T
sum=0
if num1>0
F
T
• sum= sum+ num1 %10
• num1= num1/10
num1=sum
repeat for num2
display num1 and num2
A

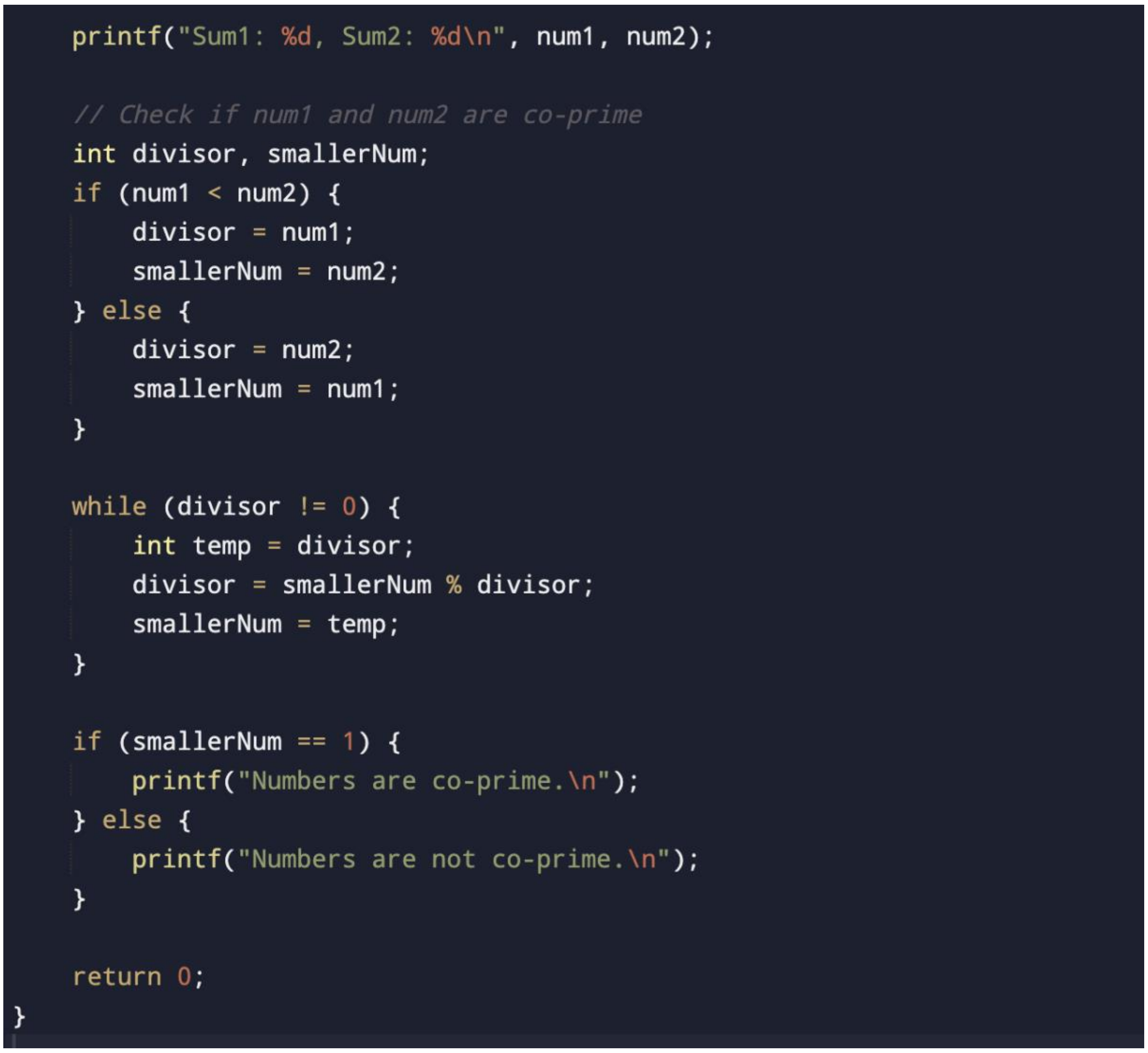


→ Input and Output:

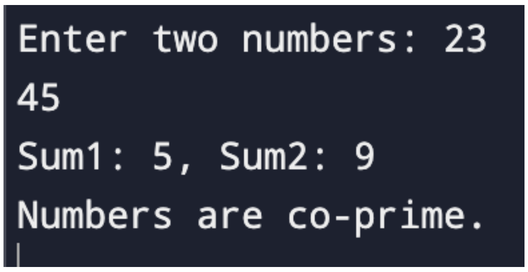
No common factors other than 1. Thus, numbers are co-primes



```c
#include <stdio.h>

int main() {
    int num1, num2;

    printf("Enter two numbers: ");
    scanf("%d %d", &num1, &num2);

    // Calculate sum of digits for num1
    while (num1 >= 10) {
        int sum = 0;
        while (num1 > 0) {
            sum += num1 % 10;
            num1 /= 10;
        }
        num1 = sum;
    }

    // Calculate sum of digits for num2
    while (num2 >= 10) {
        int sum = 0;
        while (num2 > 0) {
            sum += num2 % 10;
            num2 /= 10;
        }
        num2 = sum;
    }

    printf("Sum1: %d, Sum2: %d\n", num1, num2);
```



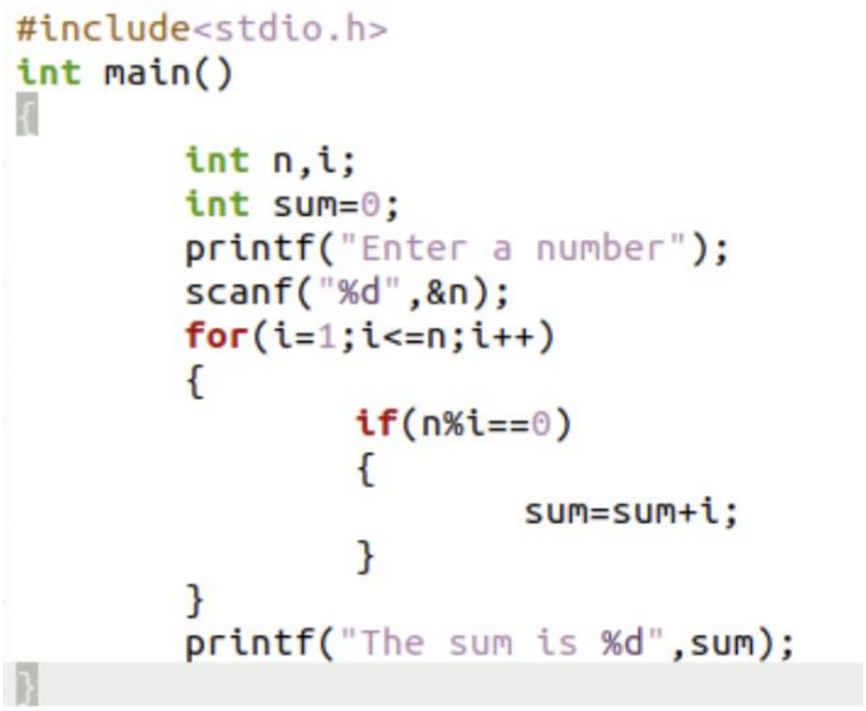
```c
    printf("Sum1: %d, Sum2: %d\n", num1, num2);

    // Check if num1 and num2 are co-prime
    int divisor, smallerNum;
    if (num1 < num2) {
        divisor = num1;
        smallerNum = num2;
    } else {
        divisor = num2;
        smallerNum = num1;
    }

    while (divisor != 0) {
        int temp = divisor;
        divisor = smallerNum % divisor;
        smallerNum = temp;
    }

    if (smallerNum == 1) {
        printf("Numbers are co-prime.\n");
    } else {
        printf("Numbers are not co-prime.\n");
    }

    return 0;
}
```



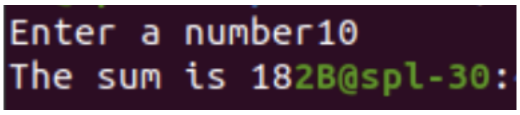
```
Enter two numbers: 23
45
Sum1: 5, Sum2: 9
Numbers are co-prime.
```

Additional programs for practice:

1. Compute the sum of all the factors of a given number

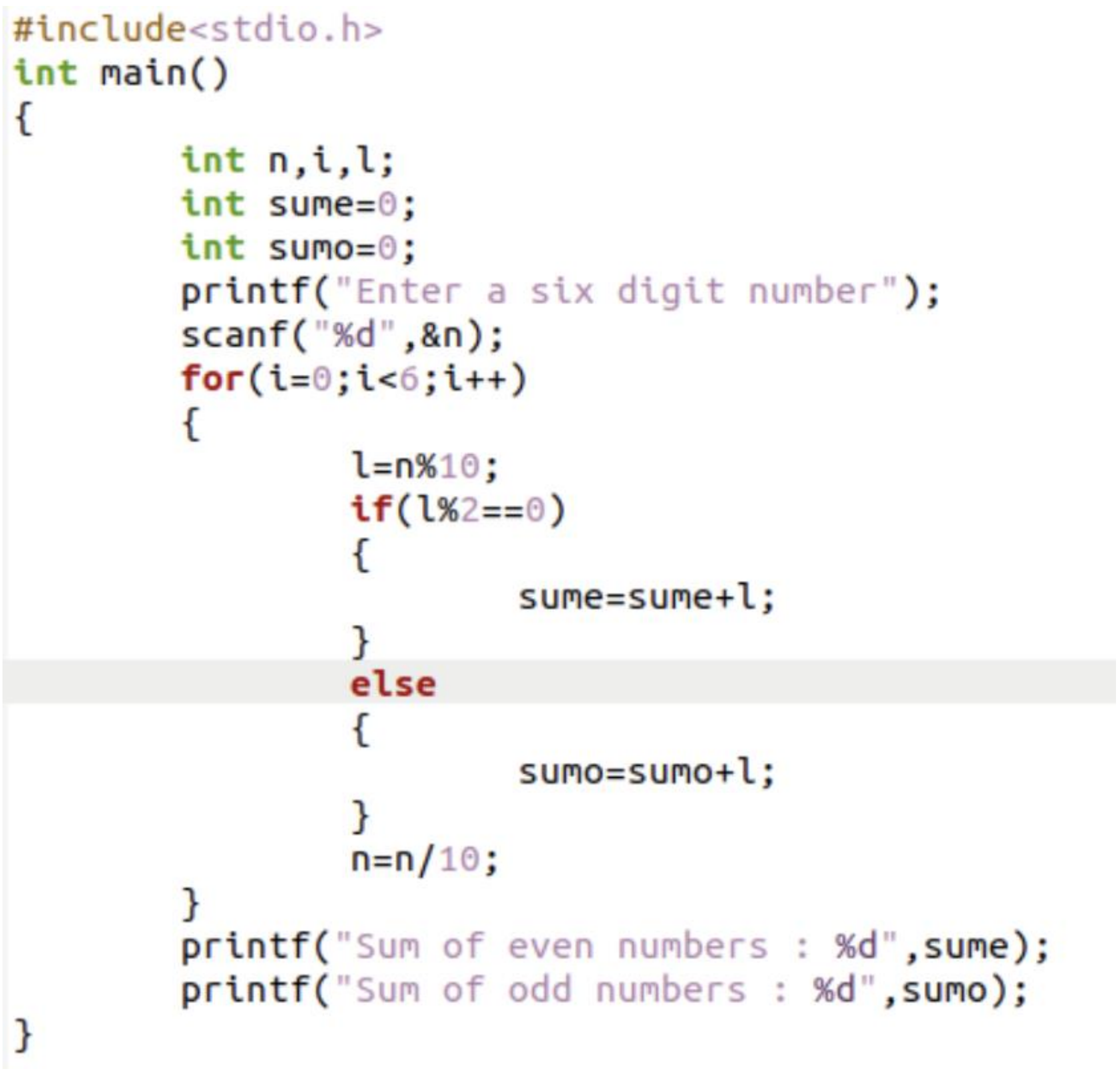


```c
#include<stdio.h>
int main()
{
	int n,i;
	int sum=0;
	printf("Enter a number");
	scanf("%d",&n);
	for(i=1;i<=n;i++)
	{
		if(n%i==0)
		{
			sum=sum+i;
		}
	}
	printf("The sum is %d",sum);
}
```

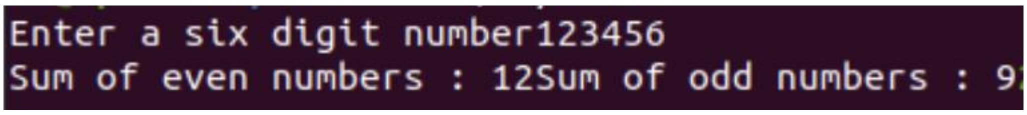


```
Enter a number10
The sum is 182B@spl-30:
```

2. Compute the difference between the sum of the odd digits and the sum of the even digits of a given six-digit number

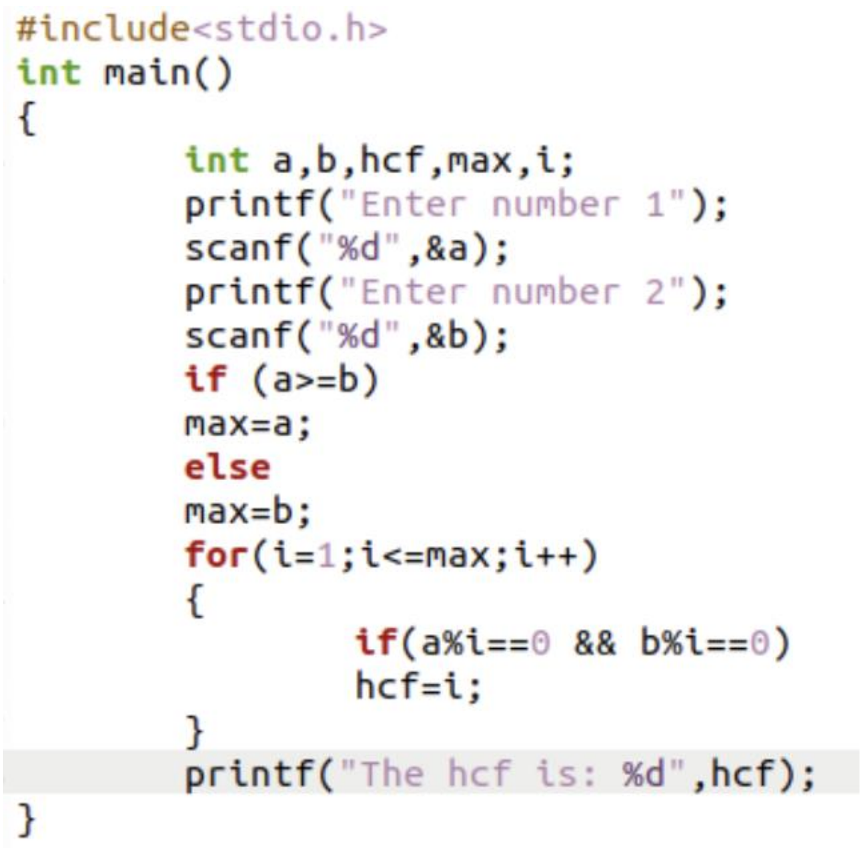


```c
#include<stdio.h>
int main()
{
        int n,i,l;
        int sume=0;
        int sumo=0;
        printf("Enter a six digit number");
        scanf("%d",&n);
        for(i=0;i<6;i++)
        {
                l=n%10;
                if(l%2==0)
                {
                        sume=sume+l;
                }
                else
                {
                        sumo=sumo+l;
                }
                n=n/10;
        }
        printf("Sum of even numbers : %d",sume);
        printf("Sum of odd numbers : %d",sumo);
}
```

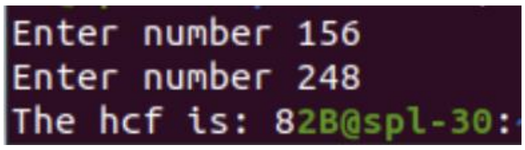


```
Enter a six digit number123456
Sum of even numbers : 12Sum of odd numbers : 9
```

3. Given two numbers, compute the highest common factor (HCF) of those two numbers

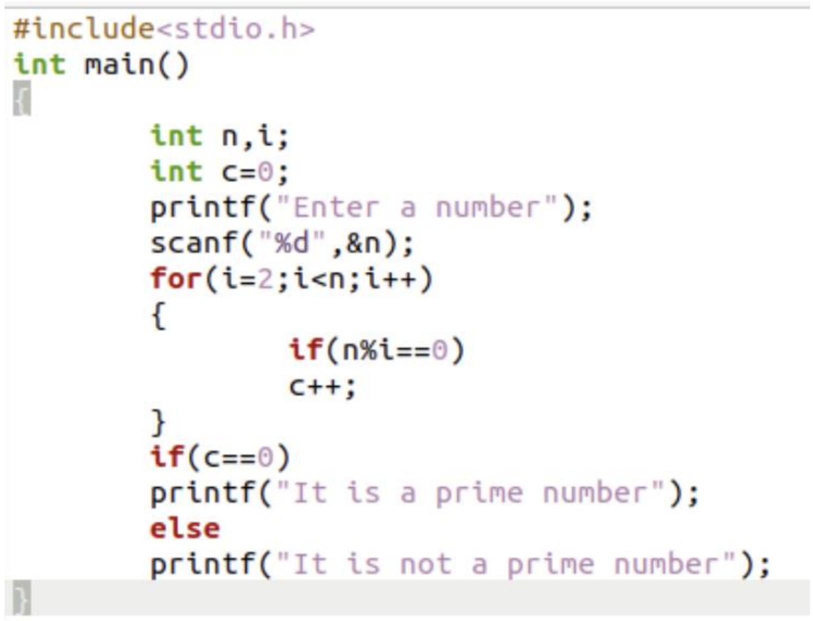


```c
#include<stdio.h>
int main()
{
	int a,b,hcf,max,i;
	printf("Enter number 1");
	scanf("%d",&a);
	printf("Enter number 2");
	scanf("%d",&b);
	if (a>=b)
	max=a;
	else
	max=b;
	for(i=1;i<=max;i++)
	{
		if(a%i==0 && b%i==0)
		hcf=i;
	}
	printf("The hcf is: %d",hcf);
}
```

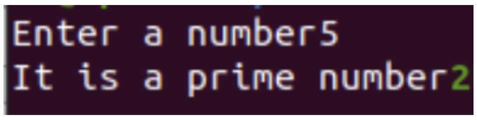


```
Enter number 156
Enter number 248
The hcf is: 82B@spl-30:
```

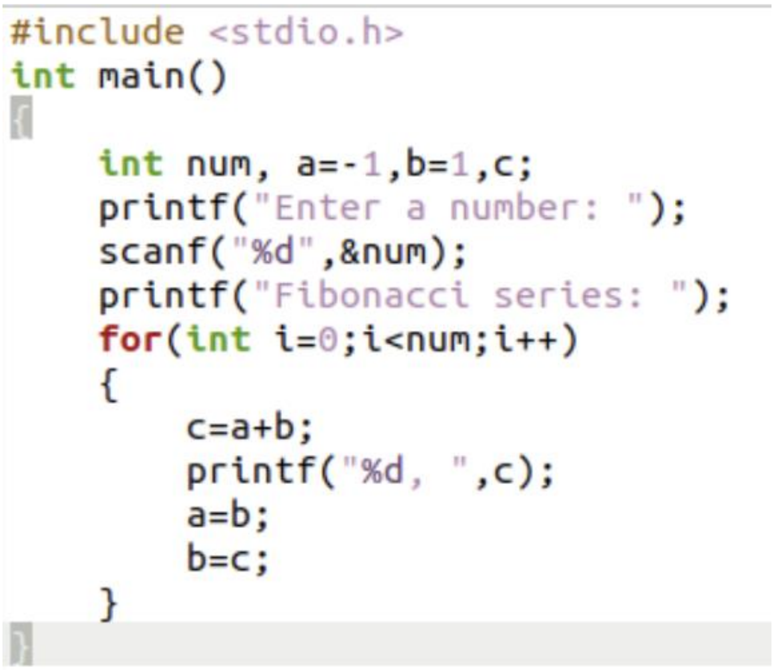
4. Check whether a given number is prime or not



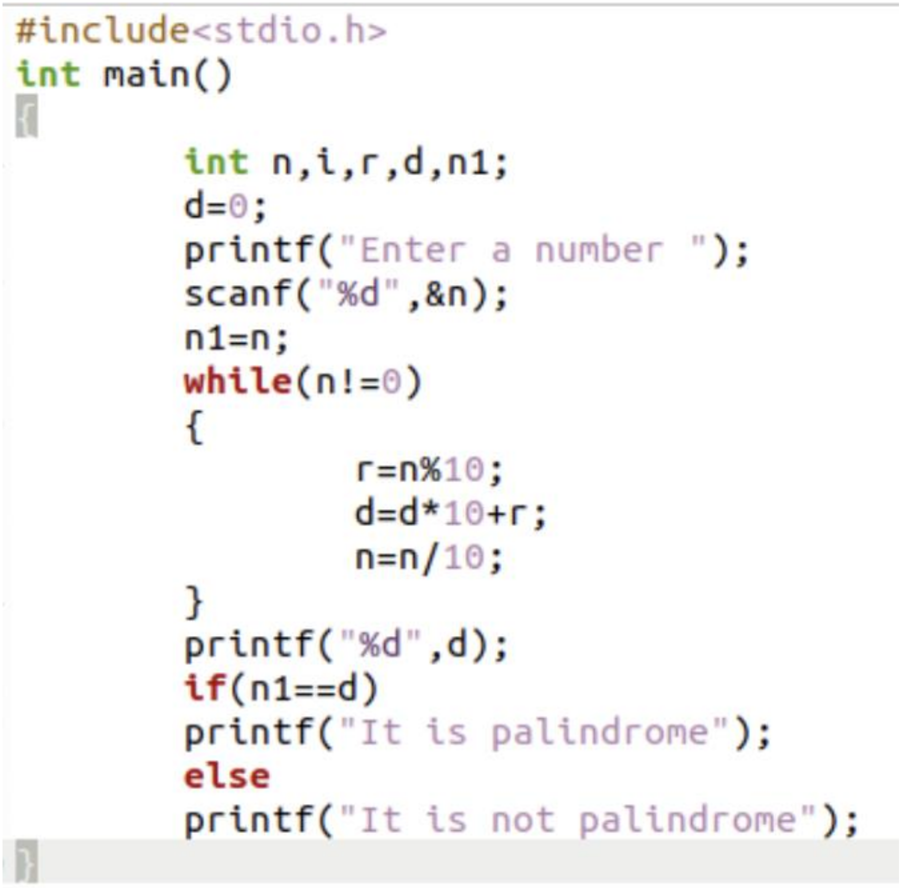
```c
#include<stdio.h>
int main()
{
	int n,i;
	int c=0;
	printf("Enter a number");
	scanf("%d",&n);
	for(i=2;i<n;i++)
	{
		if(n%i==0)
		c++;
	}
	if(c==0)
	printf("It is a prime number");
	else
	printf("It is not a prime number");
}
```

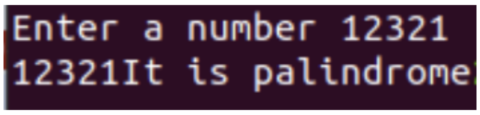

```
Enter a number5
It is a prime number2
```

5. Generate the first n Fibonacci numbers


```c
#include <stdio.h>
int main()
{
    int num, a=-1,b=1,c;
    printf("Enter a number: ");
    scanf("%d",&num);
    printf("Fibonacci series: ");
    for(int i=0;i<num;i++)
    {
        c=a+b;
        printf("%d, ",c);
        a=b;
        b=c;
    }
}
```

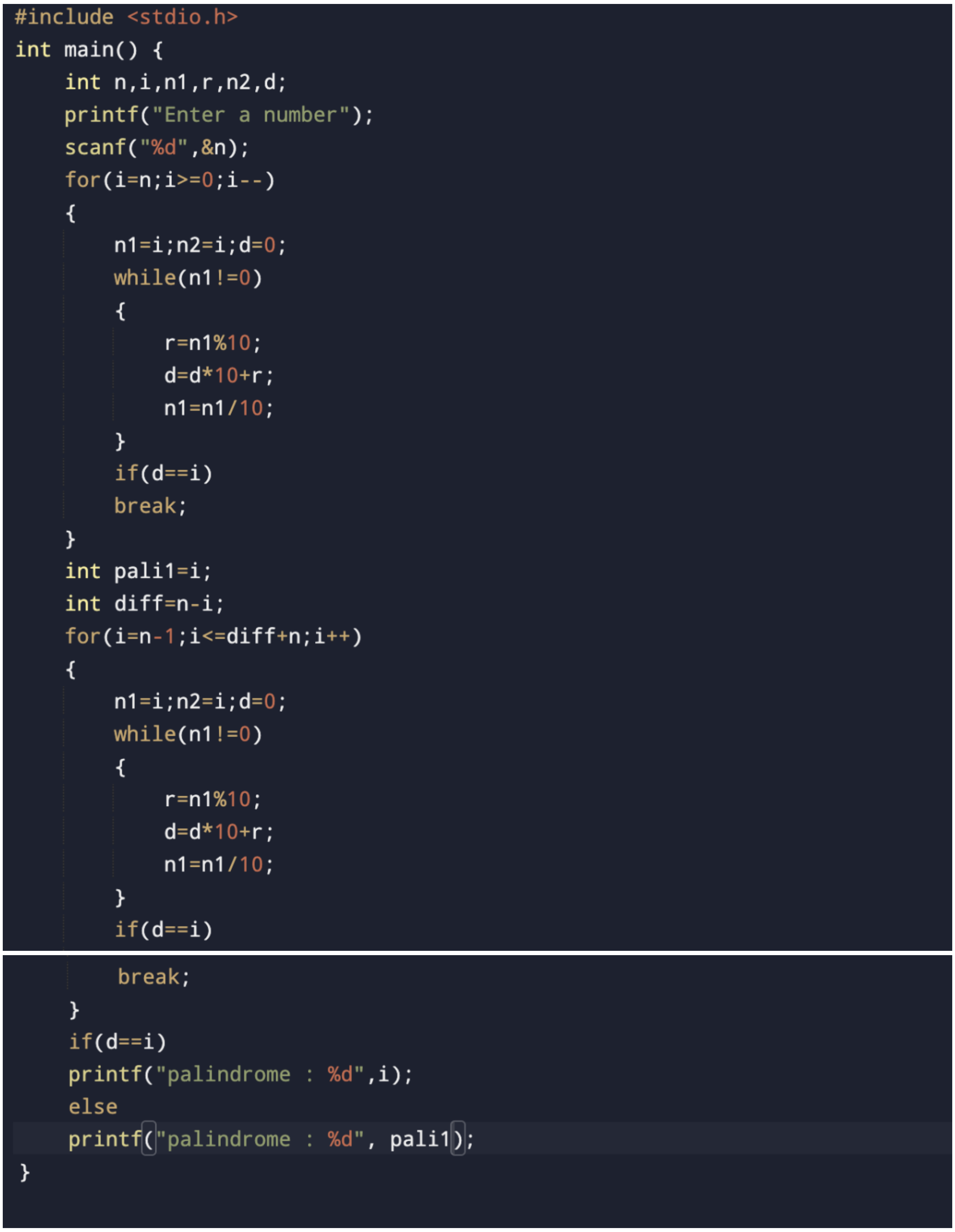
6. Check whether a given number is palindrome or not.


```c
#include<stdio.h>
int main()
{
        int n,i,r,d,n1;
        d=0;
        printf("Enter a number ");
        scanf("%d",&n);
        n1=n;
        while(n!=0)
        {
                r=n%10;
                d=d*10+r;
                n=n/10;
        }
        printf("%d",d);
        if(n1==d)
        printf("It is palindrome");
        else
        printf("It is not palindrome");
}
```

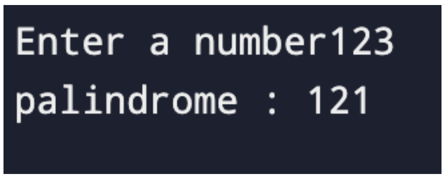


```
Enter a number 12321
12321It is palindrome
```

7. Modify Q1 to return the closest integer (not including itself) which is a palindrome. Consider the maximum number of digits for the integer to be 5. Examples: Input: n=123 Output: 121

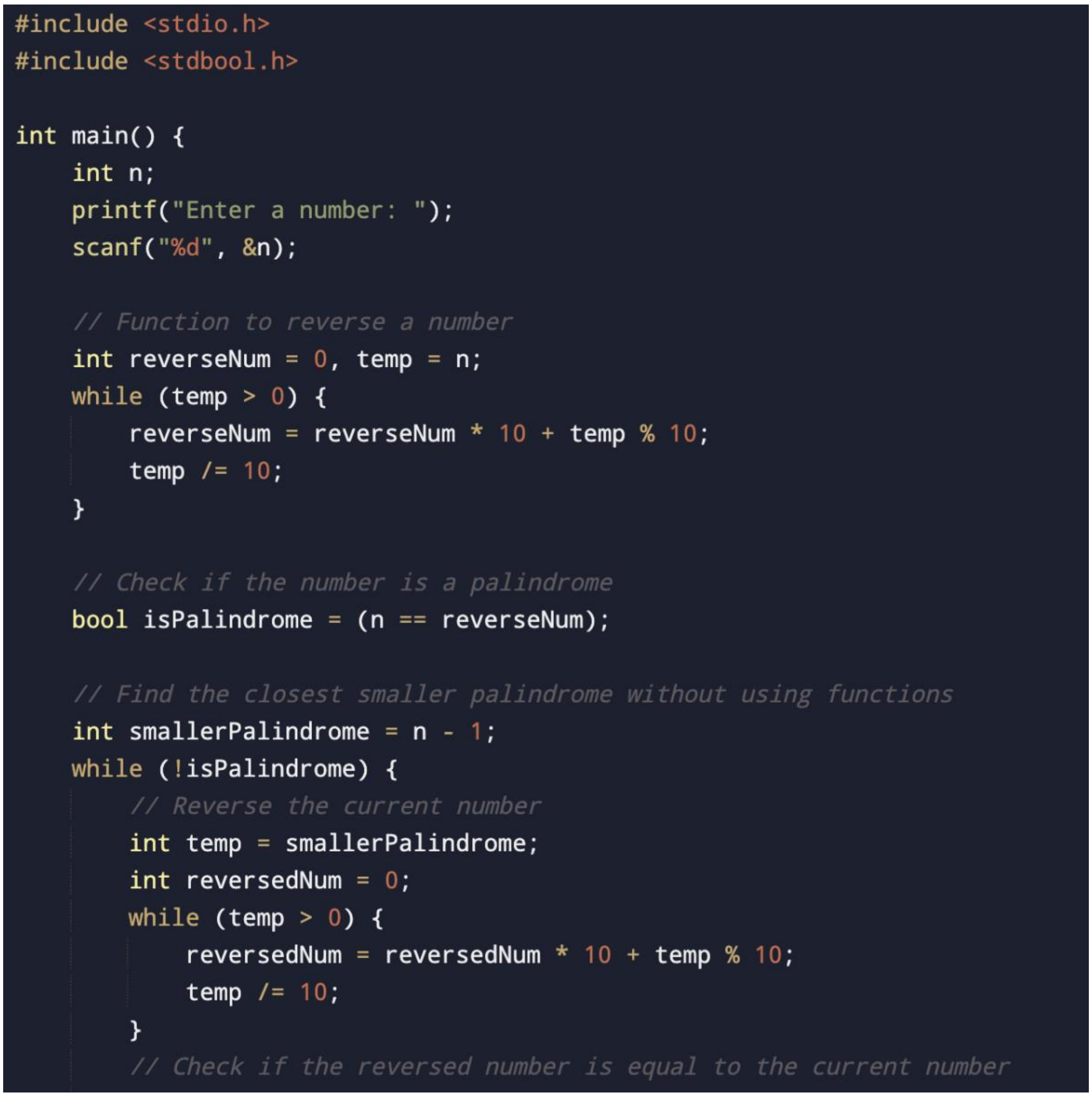


```c
#include <stdio.h>
int main() {
    int n,i,n1,r,n2,d;
    printf("Enter a number");
    scanf("%d",&n);
    for(i=n;i>=0;i--)
    {
        n1=i;n2=i;d=0;
        while(n1!=0)
        {
            r=n1%10;
            d=d*10+r;
            n1=n1/10;
        }
        if(d==i)
        break;
    }
    int pali1=i;
    int diff=n-i;
    for(i=n-1;i<=diff+n;i++)
    {
        n1=i;n2=i;d=0;
        while(n1!=0)
        {
            r=n1%10;
            d=d*10+r;
            n1=n1/10;
        }
        if(d==i)
        break;
    }
    if(d==i)
    printf("palindrome : %d",i);
    else
    printf("palindrome : %d", pali1);
}
```

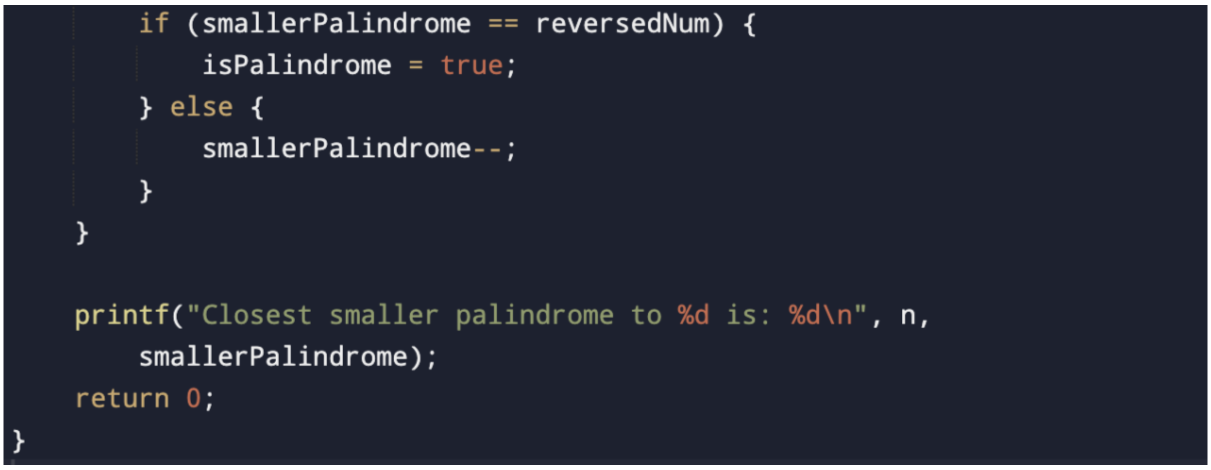


```
Enter a number123
palindrome : 121
```

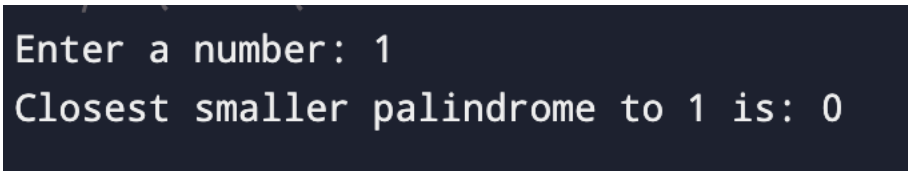
8. If there is a tie, return the smaller one. The closest is defined as the absolute difference minimized between two integers. Example: Input: n=1 Output: 0 (0 and 2 are the closest palindromes but we return the smallest which is 0)



```c
#include <stdio.h>
#include <stdbool.h>

int main() {
    int n;
    printf("Enter a number: ");
    scanf("%d", &n);

    // Function to reverse a number
    int reverseNum = 0, temp = n;
    while (temp > 0) {
        reverseNum = reverseNum * 10 + temp % 10;
        temp /= 10;
    }

    // Check if the number is a palindrome
    bool isPalindrome = (n == reverseNum);

    // Find the closest smaller palindrome without using functions
    int smallerPalindrome = n - 1;
    while (!isPalindrome) {
        // Reverse the current number
        int temp = smallerPalindrome;
        int reversedNum = 0;
        while (temp > 0) {
            reversedNum = reversedNum * 10 + temp % 10;
            temp /= 10;
        }
        // Check if the reversed number is equal to the current number
```



```
        if (smallerPalindrome == reversedNum) {
            isPalindrome = true;
        } else {
            smallerPalindrome--;
        }
    }

    printf("Closest smaller palindrome to %d is: %d\n", n,
        smallerPalindrome);
    return 0;
}
```



```
Enter a number: 1
Closest smaller palindrome to 1 is: 0
```