Text BoxSince childhood we learn even number and odd number, today we solve a problem of odd digit number. So, in this article we will learn the odd digit number and solve a problem based on these numbers. In this article, our aim is to find the smallest odd digit number which is not less than a given number.

Odd digit numbers are those numbers whose each and every digit are odd, in other words if a number formed by using only the digits 1,3,5,7, and 9 then this number is known as an odd digit number. For example, 31579373, 3175179, etc.

In this question, a number is given and we have to find the smallest odd digit number which is not less than the given number.

Let’s try to understand this question with few examples:

**Example 1:**

Suppose, the given number is 2365. So as we can see it consists of some even digits like, 2 and 6. So we need to remove it and find a number which is not less than 2365.

If we try to replace 2 with 1 and 6 with any odd digit the resulting number is smaller than 2365. So we have to replace the digit 2 with 3 and then all the digits with the smallest odd digit i.e. 1 to get the smallest odd digit number not less than 2365.

So, 3111 is our output in this example.

**Example 2:**

Let’s understand with one more example. Let the given number be 354786. We can see this number also has some even digits so we have to do some changes in the digits to get the output number. In this case we have to replace 4 with 5 to get the desired output and then the rest of the digit is set to 1.

So, 355111 is our output in this example.

**Example 3:**

Can you find the output of the given number 3571?

Let me explain, you are right the output of this number is 3571 itself because in this example there is no even digit present in the number. So, if a number with no even digit then the output is simply the given number itself.

From the above examples and explanation, it is clear to you what the problem says and how to find it mathematically. Let’s try to code the problem in c++ with explanation and algorithm.

## **Approach 1: Simple Approach(Non-recursive)**

Let’s try to solve the given problem.

### **Algorithm**

**Step 1:** Create a function smallest\_number which takes the input number and returns the output number.

**Step 2:** Create a helper function check\_digits which checks the digit of the number whether it has even digits or not.

**Step 3:** With the help of a for loop check the number to solve this problem with help of the check\_digits function.

**Step 4:** If the number has no even digits then it will return the number, otherwise it will increase the number by 1 and check again.

### **Explanation**

In this approach, we are using a simple method to solve the given problem by checking the number one by one and if the given number is not our desired output then using a for loop increases the number and again checks for the same. We are performing this until we get our desired result.

### **Program**

|  |
| --- |
| #include <iostream>  using namespace std;  int check\_digits(int n)  {  while (n) {  if ((n % 10) % 2 == 0)  return 0;  n /= 10;  }  return 1;  }  int smallest\_number(int n)  {  for (int i = n;; i++)  if (check\_digits(i))  return i;  }  int main()  {  int N = 2365;  cout << smallest\_number(N);  return 0;  } |

**Output**

|  |
| --- |
| 3111 |

We see a simple approach using a for loop to solve the problem, now let’s code it with the same logic but calling the function again and again i.e. by recursive approach.

## **Approach 2: Recursive Approach**

### **Algorithm**

**Step 1:** Create a function named smallest\_number which takes the given input and return the output number.

**Step 1:** This function uses a helper function named check\_digits.

**Step 2:** The check\_digits recursively calls itself to check the given number and return 1 if no even digits are present else return 0.

**Step 3:** Then in the smallest\_number function if the number is the smallest odd digit number then it will return the number else it again recursively calls the smallest\_number function with increasing the given number.

**Step 4:** This method repeats itself until and unless it finds the desired result for the given problem.

### **Explanation**

This approach calls its function again and again to get the smallest odd digit number. Here, the smallest\_number and check \_digits function call itself recursively to return the desired result.

### **Program**

|  |
| --- |
| #include <iostream>  using namespace std;  int check\_digits(int n)  {  if (n == 0)  return 1;  if ((n % 10) % 2 == 0)  return 0;  return check\_digits(n / 10);  }  int smallest\_number(int n)  {  if (check\_digits(n))  return n;  return smallest\_number(n + 1);  }  int main()  {  int N = 354786;  cout << smallest\_number(N);  return 0;  } |

**Output**

|  |
| --- |
| 355111 |

## **Conclusion**

In this article, we understand the odd digit number and solve the problem to find the smallest odd digit number which is not less than a given number. We learn how to find the answer logically and then solve it using algorithm. We code the problem statement in c++ to find the solution of the given problem.