We all know the permutation, i.e. arrangement in a definite order, today we solve a problem which is related to Lexicographically smallest permutation. Some of you might be confused about the term Lexicographical. So, let’s talk about it and understand the problem statement.

Lexicographical is an arrangement in which we follow the sequential order. In this question we have to find the smallest Lexicographical permutation of integer form in such a way that no digit is at its original index. It seems a little tough to understand for the first time but you will understand.

**NOTE:** The input of this problem is greater than 1 and less than 10.

Let’s try to understand with few examples:

**Example 1:**

In the first example let the input is 6. So its lexicographical order is 123456. But we have to find the number in which no digits is at its original index, in such a way that it should be the smallest. So if we switch the adjacent digit in this case we will get the desired number.

So the output in this case is 214365.

**Example 2:**

Let’s take another example and try to solve it. In this example the input is 5. So if we try to find its answer first we see its lexicographical order is 12345. Now if we switch the adjacent digit then it will be 21435. Which is not the desired result because 5 is at its original index. So, We have to move the last 3 digits in such a way that it fulfills the condition.

In this case the output is 21453.

Hope, the question is clear to everyone. Now we are going to solve this problem and code in c++.

**Approach 1: Without using recursion**

**Algorithm**

**Step 1:** Create a function smallestPermute which takes the input integer and returns the output as a string value which is a character array.

**Step 2:** We define a character array of name res to store the output.

**Step 3:** If the input number is even then we do the operation using a for loop to switch the adjacent digits to find the output string.

**Step 4:** Otherwise if the number is odd then, perform an operation to get the desired result.

**Step 5:** Next, add a null character to the end of the string and return the string as output.

**Explanation**

This is the simple way of solving this problem by calling a function smallestPermute. This function simply stores each integer value as a character in a char array and returns the string as output. This is the simple approach by which we can solve this problem using the above algorithm.

**Program**

|  |
| --- |
| #include <iostream>  using namespace std;    string smallestPermute(int n)  {  char res[n + 1];  if (n % 2 == 0) {  for (int i = 0; i < n; i++) {  if (i % 2 == 0)  res[i] = 48 + i + 2;  else  res[i] = 48 + i;  }  }  else {  for (int i = 0; i < n - 2; i++) {  if (i % 2 == 0)  res[i] = 48 + i + 2;  else  res[i] = 48 + i;  }  res[n - 1] = 48 + n - 2;  res[n - 2] = 48 + n;  res[n - 3] = 48 + n - 1;  }  res[n] = '\0';  return res;  }  int main()  {  int n = 5;  cout << smallestPermute(n);  return 0;  } |

**Output**

|  |
| --- |
| 21453 |

**Approach 2: Using Recursion**

Let’s solve the problem statement with help of recursive approach.

**Algorithm**

**Step 1:** Create a smallestPermute function to solve the problem statement.

**Step 2:** This function uses a helper function smallestPermuteHelper.

**Step 3:** smallestPermuteHelper function does its operations by recursively calling itself again and again to get the desired output.

**Step 4:** Then it arranges the output string which is used by the function smallestPermute.

**Step 5:** At last, the smallestPermute returns the output to the main function.

**Explanation**

In this approach we use a recursive approach to solve the problem statement. We recursively call the smallestPermuteHelper function inside the smallestPermute. This smallestPermuteHelper does its job to get the required string. Then with the help of the smallestPermute function we return the output in the main function as a string.

**Program**

|  |
| --- |
| #include <iostream>  using namespace std;  void smallestPermuteHelper(char\* res, int n, int i) {      if (i >= n) {          res[n] = '\0';          return;      }      if (n % 2 == 0) {          if (i % 2 == 0)              res[i] = 48 + i + 2;          else              res[i] = 48 + i;      }      else {          if (i == n - 3) {              res[i] = 48 + n - 1;              res[i+1] = 48 + n;              res[i+2] = 48 + n - 2;              smallestPermuteHelper(res, n, i+3);              return;          }          if (i % 2 == 0)              res[i] = 48 + i + 2;          else              res[i] = 48 + i;      }      smallestPermuteHelper(res, n, i+1);  }  string smallestPermute(int n) {      char res[n + 1];      smallestPermuteHelper(res, n, 0);      return res;  }    int main() {      int n = 6;      cout << smallestPermute(n);      return 0;  } |

**Output**  

|  |
| --- |
| 214365 |

**Conclusion**

In this article, we know about the term Lexicographical. Then we see a problem in this i.e. find the lexicographically smallest permutation with no digits at original Index. We learn and solve this problem and code it in c++ language.