#include <stdio.h>

#include <stdbool.h>

#include <stdlib.h>

#include <time.h>

int number\_digits[10];

bool digit\_used[10];

long long build\_partial\_number(int count) {

long long current\_value = 0;

for (int i = 0; i < count; i++) {

current\_value = current\_value \* 10 + number\_digits[i];

}

return current\_value;

}

bool find\_solution(int position) {

if (position == 10) {

return true;

}

for (int current\_digit = 0; current\_digit < 10; current\_digit++) {

if (digit\_used[current\_digit]) {

continue;

}

if (position == 0 && current\_digit == 0) {

continue;

}

number\_digits[position] = current\_digit;

digit\_used[current\_digit] = true;

bool rules\_followed = true;

if (position == 1) {

if (number\_digits[1] % 2 != 0) {

rules\_followed = false;

}

}

else if (position == 2) {

long long partial\_number = build\_partial\_number(3);

if (partial\_number % 3 != 0) {

rules\_followed = false;

}

}

else if (position == 9) {

if (number\_digits[9] != 0) {

rules\_followed = false;

}

}

if (rules\_followed) {

if (find\_solution(position + 1)) {

return true;

}

}

digit\_used[current\_digit] = false;

}

return false;

}

int main() {

srand(time(NULL));

for (int i = 0; i < 10; i++) {

digit\_used[i] = false;

}

printf("Generating a 10-digit number with unique digits and specific divisibility rules...\n");

if (find\_solution(0)) {

printf("Found a number: ");

for (int i = 0; i < 10; i++) {

printf("%d", number\_digits[i]);

}

printf("\n");

printf("\nVerification:\n");

printf(" First digit: %d \n", number\_digits);

printf(" First 2 digits: %d%d\n", number\_digits[0], number\_digits[1], (build\_partial\_number(2) % 2 == 0 ? "Yes" : "No"));

printf(" First 3 digits: %d%d%d \n", number\_digits[0], number\_digits[1], number\_digits[2], (build\_partial\_number(3) % 3 == 0 ? "Yes" : "No"));

printf(" First 10 digits: %lld \n", build\_partial\_number(10), (build\_partial\_number(10) % 10 == 0 ? "Yes" : "No"));

} else {

printf("No number found that satisfies all the rules.\n");

}

return 0;

}

#include <stdio.h>

#include <string.h>

int main() {

char s1[] = "abbcddeefceeacbbed";

char s2[] = "abbefieefceeacbijd";

int len = strlen(s1);

int start\_index = -1;

for (int i = 0; i < len; i++) {

if (s1[i] != s2[i]) {

if (start\_index == -1) {

start\_index = i;

}

} else {

if (start\_index != -1) {

for (int j = start\_index; j < i; j++) {

printf("%c", s1[j]);

}

printf(", ");

start\_index = -1;

}

}

}

if (start\_index != -1) {

for (int j = start\_index; j < len; j++) {

printf("%c", s1[j]);

}

}

printf("\n");

return 0;

}

#include <stdio.h>

// Function to find the maximum element's index in a subarray

int findMaxIndex(int arr[], int start, int end) {

int maxIdx = start;

int i;

for (i = start + 1; i <= end; i++) {

if (arr[i] > arr[maxIdx]) {

maxIdx = i;

}

}

return maxIdx;

}

void rearrangeArray(int arr[], int n) {

int start = 0;

int end = n - 1;

int temp;

int currentMaxIdx;

int i; // Declare 'i' here for compatibility with older standards

for (i = 0; start <= end; i++) {

currentMaxIdx = findMaxIndex(arr, start, end);

int maxVal = arr[currentMaxIdx];

if (i % 2 == 0) {

// Place max at start

temp = arr[start];

arr[start] = maxVal;

arr[currentMaxIdx] = temp;

start++;

} else {

// Place max at end

temp = arr[end];

arr[end] = maxVal;

arr[currentMaxIdx] = temp;

end--;

}

}

}

int main() {

int arr[] = {1, 12, 6, 5, 4};

int n = sizeof(arr) / sizeof(arr[0]);

rearrangeArray(arr, n);

// Print the rearranged array

printf("Rearranged array:\n");

int i;

for (i = 0; i < n; i++) {

printf("%d ", arr[i]);

}

printf("\n");

return 0;

}

#include<stdio.h>

int main() {

int num,r;

int rev1=0,rev2=0;

int res1=0,res2=0;

int d1=0,d2=0;

printf("Enter the no : ");

scanf("%d",&num);

int n=num;

while(num>0) {

d1=num%10;

rev1 = rev1\*10 + d1;

num=num/10;

}

printf("%d \n",rev1);

res1=n+rev1;

r=res1;

while(res1>0) {

d2=res1%10;

rev2 = rev2\*10+d2;

res1=res1/10;

}

if(r==rev2) {

printf("it is palindrome \n");

} else {

printf("Not palindrome \n");

}

}