Bài yêu cầu viết các hàm để thực hiện các yêu cầu

Yêu cầu 1: Kiểm tra n có là số nguyên tố → sử dụng hàm bool;

Kiểm tra số nguyên tố bằng cách kiểm tra xem từ 2 $\rightarrow \sqrt{n}$ $có\ số\ nào\ là\ ước\ của\ n\ không$

```
bool yc1(int n){
   if(n == 0 || n == 1) return false;
   for(int i = 2 ; i * i <= n; i++){
      if(n % i == 0) return false;
   }
   return true;
}
```

Yêu cầu 2: Tạo hàm trả về tổng các chữ số của 1 chuỗi

```
int yc2(int n){
   int sumOfDigits = 0;
   // code tinh tong cac chu so
   return sumOfDigits;
}
```

Yêu cầu 3: In tổng chữ số chẵn của n

```
int yc3(int n) {
   int sumOfEvenDigits = 0;
   // Tách các chữ số từ n và nếu nó là số chẵn thì công vào tổng sumOfEvenDigits;
   return sumOfEvenDigits;
}
```

Yêu cầu 4: In tổng chữ số của n là số nguyên tố

```
int yc4(int n) {
   int sumOfPrimeDigits = 0;
   // tách các chữ số của n và kiểm tra xem có phải là chữ số nguyên tố không và cộng vào
sumOfPrimeDigits
   return sumOfPrimeDigits;
}
```

Yêu cầu 5: In số lật ngược của n. Ví dụ 123 in 321 , 1200 → 21

```
int yc5(int n) {
   int reversedNumber = 0;
   while (n > 0) {
      int digit = n % 10; // lấy chữ số hàng đơn vị
      reversedNumber = reversedNumber * 10 + digit; // thêm vào số lật ngược
      n /= 10; // loại bỏ chữ số hàng đơn vị của n
   }
   return reversedNumber;
}
```

Yêu cầu 6: In số lượng ước riêng biệt của n là số nguyên tố (làm tương tự như phân tích thừa số ng tố).

```
int yc6(int n) {
   int countOfDistinctPrimeFactors = 0;
   // Tìm các ước của n và kiểm tra ước đó có phải là số nguyên tố không (tận dụng hàm kiểm tra yc1) nếu đúng thì tăng biến đếm lên 1
   return countOfDistinctPrimeFactors;
}
```

Yêu cầu 7: In ước nguyên tố lớn nhất của n (làm tương tự như phân tích thừa số ng tố).

```
int yc7(int n) {
   int maxPrimeFactor = -1; // khởi tạo giá trị âm để biểu thị không có kết quả
   // kiểm tra tất cả các ước là số nguyên tố của n và so sánh với giá trị maxPrimeFactor . Nếu lớn hơn thì maxPrimeFactor bằng giá trị đó
   return maxPrimeFactor;
}
```

Yêu cầu 8: Kiểm tra nếu n tồn tại ít nhất 1 số 6, nếu đúng in 1, sai in 0.

```
bool yc8(int n) {
   // duyệt qua tất cả các chữ số của n và trả về 1 nếu có chữ số nào = 6
   return false; // trả về kết quả sai sau khi duyệt tất cả mà không tìm thấy số 6
}
```

Yêu cầu 9: Kiểm tra nếu tổng chữ số của n chia hết cho 8, nếu đúng in 1, sai in 0.

```
bool yc9(int n) {
   int sumOfDigits = yc2(n); // gọi hàm yc2 đã viết để tính tổng chữ số
   if (sumOfDigits % 8 == 0) { // kiểm tra tổng chia hết cho 8
      return true; // trả về kết quả đúng

   }
   else {
      return false; // trả về kết quả sai

   }
}
```

Yêu cầu 10: Tính tổng giai thừa các chữ số của n và in ra. Ví dụ n = 123, tổng = 1! + 2! +3!

```
int yc10(int n) {
   int sumOfFactorials = 0;
   // tách các chữ số và tính giai thừa của các chữ số đó r cộng vào sumOfFactorials
   return sumOfFactorials;
}
```

Yêu cầu 11: Kiểm tra n có phải chỉ được tạo bởi 1 số hay không? Ví dụ 222, 333, 99999. Đúng in ra 1, sai in ra 0.

```
bool yc11(int n) {
   int firstDigit = n % 10; // lấy chữ số hàng đơn vị làm mẫu
   // kiểm tra các chữ số còn lại có bằng số đầu tiên không, nếu 0 trả về false
```

```
    return true; // trả về kết quả đúng khi duyệt qua tất cả mà không có số nào sai
}
```

Yêu cầu 12: Kiểm tra n có phải có chữ số tận cùng là lớn nhất hay không, tức là không có chữ số nào của n lớn hơn chữ số hàng đơn vị của nó. nếu đúng in 1, sai in 0.

```
bool yc12(int n) {
    int lastDigit = n % 10; // lấy chữ số hàng đơn vị làm mẫu
    // duyệt qua tất cả chữ số của n và nếu chữ số nào lớn hơn lastDigit thì trả về false
    return true; // trả về kết quả đúng khi ko tìm thấy số nào > lastDigit
}
```

Yêu cầu 13: In tổng lũy thừa chữ số của n với cơ số là số chữ số. ví dụ 123 thì tính 1^3+2^3+3^3.

```
long long yc13(int n) {
    long long sumOfPowers = 0;
    // k = đếm số chữ số của n;
    // duyệt qua các số 1 lần nữa và tính tổng luỹ thừa bậc k của các chữ số
    return sumOfPowers;
}
```

→ Khởi tạo hàm main thự hiện yêu cầu của bài

```
int main(){
    int n;
    cout << yc1(n) << endl;
    // ...........
}
```