1.EVEN ODD

```
void check_even_odd(int n){
    if(n%2==0){
        cout<<n<<" is Even";
        return;
    }
    cout<<n<<" is Odd";
}
```

PRIME

```
void check_prime(int n){
    for(int i=0;i<n/2;i++){
        if(n==6*i-1 || n==6*i+1){
            cout<<n<<" is a Prime";
            return;
        }
    }
    cout<<n<<" is not a Prime";
}
```

2.PALINDROME

```
void check_palindrome(string n1){
    char n[n1.size()+1];
    strcpy(n,n1.c_str());        //converting the string into a character array.
    int s=strlen(n);
    for(int i=0;i<s/2;i++){
        if(n[i]!=n[s-i-1]){
            cout<<n<<" is not a Palindrome";
            return;
     }
    }
    cout<<n<<" is a Plaindrome";
}
```

PALINDROME WITH INT

```
int reverse(int n){
    int s=0;
    int r;
    while(n!=0){
        r=n%10;
        n=n/10;
        s=(s*10)+r;
    }
    cout<<s;
    return s;
}
void check_palindrome(int n){
    int rn=reverse(n);
    if(n==rn)
        cout<<n<<" is a Palindrome";
    else
        cout<<n<<" is not a Plaindrome";
}
```

3.PERFECT NUMBER

```
void check_perfect(int n){
    int sum=0;
    for(int i=1;i<n;i++){
        if(n%i==0){
            sum+=i;
        }
    }
    if(sum==n)
        cout<<n<<" is a Perfect";
    else
        cout<<n<<" is not a Perfect number";
}
```

4.ARMSTRONG

```
int find_no_digits(int n){
    int count=0;
    while(n!=0){
        n=n/10;
        count++;
    }
    return count;
}
void check_armstrong(int n){
    int p=find_no_digits(n);
    int sum=0;
    int r;
    int temp=n;
    while(temp!=0){
        r=temp%10;
        temp=temp/10;
        sum+=pow(r,p);
    }
    if(sum==n)
        cout<<n<<" is Armstrong";
    else
        cout<<n<<" is not Armstrong";
}
```

5.STRONG

```
int fact(int n){
    int f=1;
    for(int i=1;i<=n;i++){
        f=f*i;
    }
    return f;
}
void check_strong(int n){
    int sum=0;
    int temp=n;
    while(temp!=0){
        int r=temp%10;
        temp=temp/10;
        sum+=fact(r);
    }
    if(sum==n){
        cout<<n<<" is Strong";
        return;
    }
    cout<<n<<" is not Strong";
}
```

6.COUNT THE DIGITS

```
void count_digits(int n){
    if(n==0){
        cout<<"The number has 1 digit";
        return;
    }
    int count=0;
    while(n!=0){
        n=n/10;
        count++;
    }
    cout<<"The number has "<<count<<" digit(s)";
}
```

7.FACTORIAL

```
void fact(int n){
    int fact=1;
    for(int i=1;i<=n;i++){
        fact=fact*i;
    }
    cout<<"The factorial of "<<n<<" is "<<fact;
}
```

8.FIBONACCI SERIES (Iterative)

```
void fibonacci(int n){
    int first=0;
    int second=1;
    cout<<first<<" "<<second<<" ";
    for(int i=1;i<=n-2 && n>2;i++){
        int temp=second;
        second=first+second;
        first=temp;
        cout<<second<<" ";
    }
}
```

9.Nth FIBONACCI (Iterative)

```
void fibonacci(int n){
    int first=0;
    int second=1;
    if(n==1){
        cout<<"The 1st fibonacci is "<<first;
        return;}
    if(n==2){
        cout<<"The 2nd fibonacci is "<<second;
        return;}
    for(int i=1;i<=n-2;i++){
        int temp=second;
        second=first+second;
        first=temp;
    }
    cout<<"The "<<n<<"th Fibonacci is "<<second;
}
```

10.FIBONACCI (Recursive)

```
int fibonacci(int n){
    if(n==1)
        return 0;
    if(n==2)
        return 1;
    else
        return fibonacci(n-1)+fibonacci(n-2);
}
```

SWAPPING 2 NUMBERS(without 3rd variable)

```
void swap(int a,int b){
    a=a+b;
    b=a-b;
    a=a-b;

    cout<<"First Number: "<<a<<" Second Number: "<<b;
}
```

11.SWAPPING 2 NUMBERS(with pointer)

```
void swap(int* a,int* b){
    int* temp=a;
    a=b;
    b=temp;
}
```

HELLO WORLD without semicolon

```
int main(){
    if(printf("Hello World")){}
    return 0;
}
```

12.AREA OF A TRIANGLE

```
double find_area(double a,double b,double c){
    double s=(a+b+c)/2;
    return sqrt(s*(s-a)*(s-b)*(s-c));
}
```

13.VOLUME OF A SPHERE

```
using namespace std;
double find_vol(double a){
    return (((double)4/3)*((double)22/7)*a*a*a);
}
```

14.LEAP YEAR

```
void find_leap(int n){
    if(n%4==0 && n%100!=0 || n%400==0)
        cout<<n<<" is a Leap Year";
    else
        cout<<n<<" is a not Leap Year";
}
```

15.DECIMAL TO BINARY

```
int find_binary(int a){
    int sum=0,i=1;
    while(a!=0){
        int r=a%2;
        a=a/2;
        sum+=r*i;
        i=i*10;
    }
    return sum;
}
```

16.BINARY TO DECIMAL

```
int find_decimal(int n){
    int sum=0,i=0;
    while(n!=0){
        int r=n%10;
        n=n/10;
        sum+=r*pow(2,i);
        i++;
    }
    return sum;
}
```

17.BINARY TO OCTAL

```
int decimal(int n){
    int s=0,i=0;
    while(n!=0){
        int r=n%10;
        n=n/10;
        s=s+r*pow(2,i);
        i++;
    }
    return s;
}
int find_octal(int n){
    int s=0,i=1;
    while(n!=0){
        int r=n%1000;
        r=decimal(r);
        n=n/1000;
        s=s+r*i;
        i=i*10;
    }
    return s;
}
```

18.LCM

```
int find_lcm(int a,int b){
    int low=(a<b)?a:b;
    while(1){
        if(low%a==0 && low%b==0){
            return low;
        }
        low++;
    }
}
```

19.HCF

```
int find_hcf(int a,int b){
    int sum=a+b;
    int low=(a<b)?a:b;
    int high=sum-low;
    while(low!=0){
            int temp=low;
            low=high%low;
            if(low==1)
                return 1;
            high=temp;
    }
    return high;
}
```

20.SECOND LARGEST

```
void find_largest_second(int a[],int n){
    sort(a,a+n);
    for(int i=0;i<10;i++){
        cout<<a[i]<<" ";
    }
    cout<<a[n-2];
}
```

21.LARGEST AND SMALLEST

```
void find_largest_smallest(int* a,int n){
    int l=a[0],s=a[0];
    for(int i=0;i<n;i++){
            if(l<a[i])
                l=a[i];
            if(s>a[i])
                s=a[i];
    }
    cout<<l<<" "<<s;
}
```

22.LINEAR SEARCH

```
void find_largest_smallest(int* a,int n){
    int l=a[0],s=a[0];
    for(int i=0;i<n;i++){
            if(l<a[i])
                l=a[i];
            if(s>a[i])
                s=a[i];
    }
    cout<<l<<" "<<s;
}
```

23.REMOVE DUPLICATE(with sorting)

```
void remove_duplicate(int* a,int n){
    sort(a,a+n);
    for(int i=0;i<n;i++){
        cout<<a[i]<<" ";
    }
    cout<<endl;
    vector<int> b;
    b.push_back(a[0]);
    for(int i=1;i<n;i++){
        if(a[i-1]!=a[i])
            b.push_back(a[i]);
    }
    for(int i=0;i<b.size();i++){
        cout<<b[i]<<" ";
    }
}
```

24.ROMAN TO INTEGER

```
int roman_to_integer(string n){
    char r[n.size()];
    strcpy(r,n.c_str());
    int sum=0;
    map<char,int> m{
        {'I',1},
        {'V',5},
        {'X',10},
        {'L',50},
        {'C',100},
        {'D',500}
    };
    m.insert(pair<char,int>('M',1000));
    for(int i=0;i<strlen(r);i++){
        if(i>0 && m[r[i]]>m[r[i-1]]){
            sum+=m[r[i]]-2*m[r[i-1]];
        }
        else
            sum+=m[r[i]];
    }
    return sum;
}
```

25.INTEGER_TO_ROMAN

```
vector<string> integer_to_roman(int n){
    vector<string> v;
    string unit[]={"","I","II","III","IV","V","VI","VII","VIII","IX","X"};
string ten[]={"","X","XX","XXX","XL","L","LX","LXX","LXXX","XC","C"};
string hundred[]={"","C","CC","CCC","CD","D","DC","DCC","DCCC","CM","M"};
string
thousands[]={"","M","MM","MMM","MMMM","MMMMM","MMMMMM","MMMMMMM","MMMMMMMM","
MMMMMMMMM","MMMMMMMMMM"};
    v.push_back(thousands[n/1000]);
    v.push_back(hundred[(n%1000)/100]);
    v.push_back(ten[(n%100)/10]);
    v.push_back(unit[(n%10)]);
    return v;
}
```