Fibonacci numbers

The **Fibonacci numbers**, commonly denoted `F(n)` form a sequence, called the **Fibonacci sequence**, such that each number is the sum of the two preceding ones, starting from `0` and `1`. That is,

```
F(0) = 0, F(1) = 1

F(n) = F(n - 1) + F(n - 2), for n > 1.
```

Given `n`, calculate `F(n)`.

Example 1:

```
Input: n = 2

Output: 1

Explanation: F(2) = F(1) + F(0) = 1 + 0 = 1.
```

Example 2:

```
Input: n = 3

Output: 2

Explanation: F(3) = F(2) + F(1) = 1 + 1 = 2.
```

Example 3:

```
Input: n = 4

Output: 3

Explanation: F(4) = F(3) + F(2) = 2 + 1 = 3.
```

Program

```
#include <iostream>

using namespace std;

int fib(int n) {

       if(n==0)

          return 0

       if (n==1)

          return 1;

       int a=0,b=1,c;

       for(int i=1;i>n;i++)

       {

          c=a+b;

          a=b;

          b=c;

       }

       return c;

}

int main () {

  int number = 5;  // local variable declaration:

  int res;

  // calling a function to get Fibonacci Number.

  res = fib(number,res);

  cout << "output : " << res << endl;

  return 0;

}
```

Happy Number

Write an algorithm to determine if a number `n` is happy.

A **happy number** is a number defined by the following process:

- Starting with any positive integer, replace the number by the sum of the squares of its digits.
- Repeat the process until the number equals 1 (where it will stay), or it **loops endlessly in a cycle** which does not include 1.
- Those numbers for which this process **ends in 1** are happy.

Return `true` *if* `n` *is a happy number, and* `false` *if not.*

Example 1:

```
Input: n = 19

Output: true

Explanation:

1² + 9² = 82

8² + 2² = 68

6² + 8² = 100

1² + 0² + 0² = 1
```

Example 2:

```
Input: n = 2

Output: false
```

Program

```
#include<iostream>

using namespace std;

bool isHappy(int n) {

         int num = n;

        bool ans;

        int sum=0;




       while(num == 1) {

            if(num==89)

            {

                return false;

            }


            while(num != 0){

                int rem = num%10;

                sum = sum + (rem*rem);

                num = num/10;

            }

         num = sum;

           sum=0;

        }


           return true;
```

```
  }
int main (){
  int num = 19;
  bool res = isHappy(num,1);
  if(res){
    cout<<"true"<<endl;
  }
  else
  {
    cout<<"false"<<endl;
  }
  return 1;
}
```

Merge Sort

Output

```
Enter the number of elements: 6
Enter elements:
14 20 78 98 20 45
Array before Sorting: 14 20 78 98 20 45
Array after Sorting: 14 20 20 45 78 98
```

Program

```
#include<iostream>
using namespace std;
void swapping(int &a, int &b) {     //swap the content of a and b
  int temp;
  temp = a;
```

```
   a = b;
   b = temp;
}
void display(int *array, int size) {
   for(int i = 0; i<size; i++)
      cout << array[i] << " ";
   cout << endl;
}
void merge(int *array, int l, int m, int r) {
   int i, j, k, nl, nr;
   //size of left and right sub-arrays
   nl = m-l+1; nr = r-m;
   int larr[nl], rarr[nr];
   //fill left and right sub-arrays
   for(i = 0; i<nl; i++)
      larr[i] = array[l+i];
   for(j = 0; j<nr; j++)
      rarr[j] = array[m+1+j];
   i = 0; j = 0; k = l;
   //marge temp arrays to real array
   while(i < nl || j<nr) {
      if(larr[i] <= rarr[j]) {
         array[k] = larr[i];
         i++;
      }else{
         array[k] = rarr[j];
         j++;
      }
      k++;
   }
   while(i<nl) {       //extra element in left array
      array[k] = larr[i];
```

```
    i++; k++;
  }
  while(j<nr) {     //extra element in right array
    array[k] = rarr[j];
    j++; k++;
  }
}
void mergeSort(int *array, int l, int r) {
  int m;
  if(l < r) {
    int m = l+(r-l)/2;
    // Sort first and second arrays
    mergeSort(array, l, m,l);
    mergeSort(array, m+1, r);
    merge(array, l, m, r);
  }
}
int main() {
  int n;
  cout << "Enter the number of elements: ";
  cin >> n;
  int arr[n];     //create an array with given number of elements
  cout << "Enter elements:" << endl;
  for(int i = 0; i<n; i++) {
    cin >> arr[i];
  }
  cout << "Array before Sorting: ";
  display(arr, n);
  mergeSort(arr, 0, n-1);     //(n-1) for last index
  cout << "Array after Sorting: ";
  display(arr, n);
}
```