**Tanvir Anjom Siddique Alvi**

**RUET-CSE19**

//Add two numbers in C++

#include<iostream>

using namespace std;

/\*int main()

{

    long long x,y;

    cin>>x>>y;

    cout<<x+y<<endl;

    return 0;

}

\*/

int main()

{

    int a=0;

    int b=0;

    int sum=0;

    std::cin>>a;

    std::cin>>b;

    sum=a+b;

    std::cout<<sum;

    return 0;

}

/\*

There will be an array of n elements.Show the maximum product of 2 elements of the array.

\*/

#include<iostream>//for cin/cout/ others

#include<cstdlib>//for using rand() function

#include<vector>//for using vector

#include<ctime>//time(0)

using std::vector;

using std::cin;

using std::cout;

long long MaxPairWiseProduct(const vector<int>& numbers)//vector pass by reference

{

    long long result=0;

    int n=numbers.size();//Determine vector size.

//…………………………………………………………………Go for all possible value………………………………………………………………………….

    for(int i=0;i<n;++i)

    {

        // for(long long j=0/i;j<n;j++) is wrong as compare i and after i;

        for(int j=i+1;j<n;++j)

        {

            if((long long)numbers[i]\*numbers[j]>result)//type casting is necessary for multiplication

            {

                result=(long long)numbers[i]\*numbers[j];

            }

        }

    }

    return result;

}

//……………………………………………………………………………………………………………………………………………………………………………………………………………………………..//

long long MaxPairWiseProductFAST(const vector<int>& numbers)

{

    int n=numbers.size();

    int max\_index1=-1;

    for(int i=0;i<n;++i)

        if((max\_index1==-1)||(numbers[i]>numbers[max\_index1]))

            max\_index1=i;

    int max\_index2=-1;

    for(int j=0;j<n;++j)

        //if((numbers[j]!=numbers[max\_index1])&&(max\_index2==-1)||(numbers[j]>numbers[max\_index2]))

        if( (j!=max\_index1)&&( (max\_index2==-1)||(numbers[j]>numbers[max\_index2]) ) )

            max\_index2=j;

//not same element(index) but can be of equal value

    //if((numbers[j]]!=numbers[max\_index1])&&(max\_index2==-1)||(numbers[j]>numbers[max\_index2]))

    //is wrong as 4 5 9 8 9, output should be 9\*9;but it will show 9\*8

    //which is wrong if 2 or more max numbers(9=9,9 is 2 times here) are there;

    //so instead of value use index

    return ((long long)numbers[max\_index1]\*numbers[max\_index2]);

}

int main()

{

    /\*

    ///...........STRESS TEST to DEBUG a program................

    srand(time(0));

    while(true)

    {

        int n;

        n=rand()%10+2;//atleat 2 elements needed for pair

        vector<int>numbers(n);

        for(int i=0;i<n;++i)

        {

            numbers[i]=rand()%1000+1;

            cout<<numbers[i]<<" ";

        }

        cout<<std::endl;

        long long result\_1=MaxPairWiseProduct(numbers);

        long long result\_2=MaxPairWiseProductFAST(numbers);

        cout<<result\_1<<' '<<result\_2<<" ";

        if(result\_1!=result\_2)

        {

            cout<<" Error\n";

            break;

        }

        cout<<" ok"<<std::endl;

    }\*/

    //.......................

    int n;

    //cout<<"Enter n:"<<std::endl;

    cin>>n;

    vector<int>numbers(n);

    for(int i=0;i<n;++i)

    {

        cin>>numbers[i];

    }

//    long long result=MaxPairWiseProduct(numbers);

long long result= MaxPairWiseProductFAST(numbers);

    cout<<result<<"\n";

    //long long result = MaxPairWiseProduct (vector<int>(1000,5));

    //pass a vector size 1000 and each element is 5;

    //cout<<result<<"\n";

    return 0;

}

//learn??BINET'S formula

//find the relationship between the sum of Fibonacci numbers

//f(n+1)=f(n)+f(n-1)

//f(n-1)=f(n+1)-f(n)

// f(0)+f(1)+f(2)+f(3)+f(4)+......f(n-1)=f(n+1)-f(1)

// Sum(1 to n-1 th)=Fib(n+1)-1;

//..............................

// sum(1 to n 'th)= Fib(n+2)-1; //BINET's Formula//Fib[2]=1 /whatever Fib[1]=0/1(0112/1123)

// Sum Of 1st n fib numbers (Fib[0]+Fib[1]+.....+Fib[n]) = Fib[n+2]-Fib[2]

//.............................proved by mathematical Induction

#include <iostream>

#include<ctime>

#include<cstdlib>

#include <cmath>

#include <algorithm>

#include <vector>

typedef long long int ll;

using namespace std;

ll Periodic(ll m) {

ll a = 0, b = 1, c = a + b;

for (int i = 0; i < m\*m; i++) {

c = (a + b) % m;

a = b;

b = c;

if (a == 0 && b == 1)

return i + 1;

}

}

int Solve(ll n, ll m) {

long long remainder = n % Periodic(m);

long long first = 0;

long long second = 1;

long long res = remainder;

for (int i = 1; i < remainder; i++) {

res = (first + second) % m;

first = second;

second = res;

}

return res % m;

}

// Sum Of 1st n fib numbers (Fib[0]+Fib[1]+.....+Fib[n]) = Fib[n+2]-Fib[2]

ll Huge\_Fibonacci(ll n) {

int Last\_Digit\_Of\_nPlus2 = Solve(n + 2, 10);//fib(n+2)%10

int Last\_Digit\_Of\_2 = Solve(2, 10);//fib(2)%10

if (Last\_Digit\_Of\_nPlus2 >= Last\_Digit\_Of\_2)

return (Last\_Digit\_Of\_nPlus2 - Last\_Digit\_Of\_2);

else

return ((10 + Last\_Digit\_Of\_nPlus2) - Last\_Digit\_Of\_2);

//make Fib(n+2)%10 Larger so that the sum become +ve

}

int main() {

long long from,to;

std::cin >>from>>to;

cout << ( (Huge\_Fibonacci(to)+10-Huge\_Fibonacci(from-1))%10) << endl;//Fastest algo

//sum(0-to)+10 - sum(0 to from-1) = from\_to all fib sum

}