```
/*
Given an array of integers, every element appears twice
except for one. Find that single
one.
sample input: { 1, 1, 2, 3, 2, 4, 5, 4, 5 }
sample input: 3
*/

#include<iostream>
using namespace std;
#define loop(i,n) for(int i=0;i<n;i++)

int main ()
{
  int size;
  cout << "Enter size of array:" << endl;
  cin >> size;
  int *arr = new int[size];
  cout << "Enter elements of array:" << endl;
  loop(i,size) cin >> arr[i];
  int result = arr[0];
  for (int j = 1; j < size; j++) result ^= arr[j];
  cout << "Output: " << result;
  delete arr;
}
```

```
/*
Given an integer array A of N integers, find the pair of
integers in the array which have
minimum XOR value. Report the minimum XOR value.

sample input: { 9, 5, 3 }
sample input: 6
*/

#include<iostream>
#include<algorithm>
using namespace std;

#define MAX 100000
#define loop(i,n) for(int i=0;i<n;i++)

int main()
{
   int size;
   cout << "Enter size of array:" << endl;
   cin >> size;
   int *arr = new int[size];
   cout << "Enter elements of array:" << endl;
   loop(i,size) cin >> arr[i];

   sort(arr, arr + size);

   int minXor = MAX;
   int val = 0;

   for (int i = 0; i < size - 1; i++) {
      minXor = min(minXor, arr[i] ^ arr[i + 1]);
   }

   cout <<"Output: "<<minXor << endl;


   delete arr;
```

```
/*
Given an integer A representing the square blocks. The
height of each square block is 1.
The task is to create a staircase of max height using
these blocks.
The first stair would require only one block, the second
stair would require two blocks
and so on. Find and return the maximum height of the
staircase

sample input: 5
sample input: 2
*/

#include<iostream>
using namespace std;

#define loop(i,size) for(int i=1;;i++)
#define output(blocks) cout <<"Output: "<< blocks <<
endl;

int main()
{
   int size, blocks = 0;
   cout << "Enter number of square blocks:" << endl;
   cin >> size;

   loop(i,size){
      size -= i;
      if(size < 0){
            output(blocks)
            break;
            }
            else blocks += 1;

      }
}
```

```
/*
Given an integer A, find and return the number of paths
in a grid of size (A x A) that
starts from (1, 1) and reaches (A, A) without crossing the
major diagonal.

sample input: 3
sample input: 2
*/

#include <iostream>
#include <cstring>

using namespace std;

#define ALLOC(size,dp) memset(dp,0,size*sizeof(*dp));

int main()
{
      int size;
      cout << "Enter number of rows and columns:" <<
endl;
   cin >> size;

   int *dp = new int[size];

   ALLOC(size,dp)

   dp[0] = 1;

   for (int i = 1; i < size; i++) {
      for (int j = 1; j < size; j++) {
        if(j<=i) dp[j] += dp[j - 1];
              else break;
      }
   }

      cout << "Output: "<<dp[size-1];
      delete dp;
}
```

```
/*
Given an integar A. Compute and return the square root of A.
If A is not a perfect square, return floor(sqrt(A)). DO NOT USE SQRT FUNCTION FROM STANDARD LIBRARY

sample input: 11
sample input: 3
*/

#include<bits/stdc++.h>
using namespace std;

int main()
{
      int num;
      cout << "Enter number: " << endl;
   cin >> num;

      if (num == 0 || num == 1) cout << "Output: "<<num;

      else
      {
            int start = 1, end = num, ans;
            while (start <= end)
            {
                  int mid = (start + end) / 2;

                  if (mid*mid == num)
                        return mid;


                  if (mid*mid < num)
                  {
                        start = mid + 1;
                        ans = mid;
                  }
                  else end = mid-1;
            }
```