```
/*
Q1. Given an array of N integers. Your task is to print the sum of all of the integers.
Example 1:
Input:
4
1 2 3 4
Output:
10
Example 2:
Input:
6
5 8 3 10 22 45
Output:
93

*/

#include <iostream>
#include <vector>


void printOutput (std::vector <int> &);
int  performOperation (std::vector<int> &, int&);

void printOutput (std::vector <int> &received) {
    for (const auto& i: received) {
        std::cout <<  i  << " ";
    }
    std::cout << std::endl;
}

int performOperation (std::vector<int>& received, int& vectorSize ) {
 printOutput(received);
 int result {0};
 int i {0};
 while(i<vectorSize) {
  std::cout << received[i] << "\t";
  result += received[i];
  ++i;
 }
 std::cout << std::endl;
 return result;
}


int main() {

    std::vector <int> vec1{};
    std::vector <int> vec2{5,8,3,10,22,45};

    //int n{0}; // size of vector
    //std::cout << "Enter size of vector \n";
    //std::cin >> n;

    //pushIntoVector(n);
```

```
    vec1.push_back(1);
    vec1.push_back(2);
    vec1.push_back(3);
    vec1.push_back(4);

    printOutput(vec1);
    int sizeofVector1=vec1.size();
    std::cout << sizeofVector1 << std::endl;
    std::cout << "================================================================
==\n";
    std::cout << performOperation (vec1,sizeofVector1) << std::endl;

    printOutput(vec2);
    int sizeofVector2=vec2.size();
    std::cout << sizeofVector2 << std::endl;
    std::cout << "================================================================
==\n";
    std::cout << performOperation (vec2,sizeofVector2) << std::endl;


    return (0);
}

/*
Q2. Given an array A[] of N integers and an index Key. Your task is to print the element present at
index key in the array.
Example 1:
Input:
5 2
10 20 30 40 50
Output:
30
Example 2:
Input:
7 4
10 20 30 40 50 60 70
Output:
50
*/



#include <iostream>
#include <vector>

void printOutput (std::vector <int> &);
int  performOperation (std::vector<int> &, int&, int&);

void printOutput (std::vector <int> &received) {
 for (const auto& i: received) {
  std::cout <<  i  << " ";
 }
 std::cout << std::endl;
}
```

```
int performOperation (std::vector<int>& received, int& vectorSize, int& key) {
 printOutput(received);
 int result {0};
 int i {0};
 std::cout << "Size vector " << vectorSize << std::endl;
 std::cout << "Key " << key << std::endl;
 //std::cout << received[key] << std::endl;
 if (key > vectorSize || key < 0 ) {
  //std::string returnType {"Error key cannot be greater than array size "};
  exit (-1);
 }

 while(i<vectorSize && i<= key) {
  if (received[key] == received[i]) {
   std::cout << "Key found " << received[key] << " " << received[i] << "\t";
   result = received[key];
  }
  ++i;
 }
 std::cout << std::endl;
 return result;
}



int main() {
 std::vector <int> vec1{};
 std::vector <int> vec2{10,20,30,40,50,60,70};

     //pushIntoVector(n);
     vec1.push_back(10);
     vec1.push_back(20);
     vec1.push_back(30);
     vec1.push_back(40);
 vec1.push_back(50);

     printOutput(vec1);
     int sizeofVector1=vec1.size();
 int key1{2};
     std::cout << sizeofVector1 << std::endl;
 std::cout << "================================================================
=\n";
     std::cout << "output 1: " << performOperation (vec1,sizeofVector1, key1) << std::endl;

     printOutput(vec2);
     int sizeofVector2=vec2.size();
 int key2 {4};
     std::cout << sizeofVector2 << std::endl;
     std::cout << "================================================================
===\n";
     std::cout << "output 2: " << performOperation (vec2,sizeofVector2,key2) << std::endl;

 //check key at 0
 int key3{0};
     std::cout << "================================================================
```

```
===\n";
 std::cout << "output 3: " << performOperation (vec2,sizeofVector2,key3) << std::endl;

    //check key at -ve index
 int key4{-1};
    std::cout << "================================================================
===\n";
 std::cout << "output 4: " << performOperation (vec2,sizeofVector2,key4) << std::endl;

 //check key out of bound
 int key5{100};
    std::cout << "================================================================
===\n";
 std::cout << "output 5: " << performOperation (vec2,sizeofVector2,key4) << std::endl;

 //check key and vector size same
 int key6{7};
    std::cout << "================================================================
===\n";
 std::cout << "output 6: " << performOperation (vec2,sizeofVector2,key4) << std::endl;
    return (0);
}

/*
Q3. Given an sorted array A of size N. Find number of elements which are less than or equal to given
element X.
Example 1:
Input:
N = 6
A[] = {1, 2, 4, 5, 8, 10}
X = 9
Output:
5
Example 2:
Input:
N = 7
A[] = {1, 2, 2, 2, 5, 7, 9}
X = 2
Output:
4
*/


#include <iostream>
#include <vector>

void printOutput (std::vector <int> &);
int  performOperation (std::vector<int> &, int&, int&);

void printOutput (std::vector <int> &received) {
 for (const auto& i: received) {
  std::cout <<  i  << " ";
 }
 std::cout << std::endl;
}
```

```
int performOperation (std::vector<int> &received, int& vectorSize, int& key) {
 int i {0},count{0}, result {0};
    std::cout << "Size vector " << vectorSize << std::endl;
 std::cout << "Key " << key << std::endl;
    while(i<vectorSize) {
        if(received[i] <= key) {
            result = ++count;
  }
  if (received[i] == received[i+1] << key) {
            result = ++count;
        }
  ++i;
 }
 return result;
}

int main() {
 std::vector <int> vec1{};
 std::vector <int> vec2{1,2,2,2,5,7,9};

    //pushIntoVector(n);
    vec1.push_back(1);
    vec1.push_back(2);
    vec1.push_back(4);
    vec1.push_back(5);
 vec1.push_back(8);
    vec1.push_back(10);


    printOutput(vec1);
    int sizeofVector1=vec1.size();
 int key1{9};
    std::cout << sizeofVector1 << std::endl;
    std::cout << "============================================================
===\n";
    std::cout << "output 1: " << performOperation (vec1,sizeofVector1, key1) << std::endl;

    printOutput(vec2);
    int sizeofVector2=vec2.size();
 int key2{2};
    std::cout << sizeofVector2 << std::endl;
    std::cout << "============================================================
===\n";
    std::cout << "output 1: " << performOperation (vec1,sizeofVector2, key2) << std::endl;
    return (0);
}

/*
Q4. You are given an array A of size N. You need to print elements of A in alternate order (starting
from index 0).
Example 1:
Input:
N = 4
A[] = {1, 2, 3, 4}
```

Output:
1 3
Example 2:
Input:
N = 5
A[] = {1, 2, 3, 4, 5}
Output:
1 3 5
*/

```
#include <iostream>
#include <vector>

void printOutput (std::vector <int> &);
void  performOperation (std::vector<int> &, int&);

void printOutput (std::vector <int> &received) {
    for (const auto& i: received) {
        std::cout <<  i  << " ";
    }
    std::cout << std::endl;
}

void  performOperation (std::vector<int> & received, int& vectorSize){
      int i {1};
      std::vector <int> outputVector{};
     std::cout << "Size vector " << vectorSize << std::endl;
        while (i<=vectorSize) {
            if (i % 2 != 0) {
                //std::cout << "Odd detected " << i << std::endl;
                //received[i];
                outputVector.push_back(i);
            }
            ++i;
        }
        printOutput(outputVector);
        return;
}

int main () {
     std::vector <int> vec1{};
     std::vector <int> vec2{1,2,3,4,5};

     //pushIntoVector(n);
     vec1.push_back(1);
     vec1.push_back(2);
     vec1.push_back(3);
     vec1.push_back(4);

     printOutput(vec1);
     int sizeofVector1=vec1.size();
     std::cout << sizeofVector1 << std::endl;
     std::cout << "==========================================================================
===\n";
     std::cout << "output 1: \n";
```

```
     performOperation (vec1,sizeofVector1);

    printOutput(vec2);
    int sizeofVector2=vec2.size();
    std::cout << sizeofVector2 << std::endl;
    std::cout << "================================================================
===\n";
    std::cout << "output 1: \n";
     performOperation (vec1,sizeofVector2);
    return (0);
}

/*
Q5. Given an array Arr of N positive integers. Your task is to find the elements whose value is equal
to that of its index value ( Consider 1-based indexing ).
Example 1:
Input:
N = 5
Arr[] = {15, 2, 45, 12, 7}
Output: 2
Explanation: Only Arr[2] = 2 exists here

Example 2:
Input:
N = 1
Arr[] = {1}
Output: 1
Explanation: Here Arr[1] = 1 exists
*/


#include <iostream>
#include <vector>

void printOutput (std::vector <int> &);
int performOperation(std::vector <int>&, int&);

void printOutput (std::vector <int> &received) {
   for (const auto& i: received) {
      std::cout <<  i  << " ";
   }
   std::cout << std::endl;
}
int performOperation(std::vector <int>& received, int& sizeVector) {
   int i{}, output {};
   for (i=1;i<=sizeVector;++i) {
      std::cout << received [i-1] << " " << i << " \n ";
      if (received [i-1] == i ) {
         output = i;
         break;
      }
   }
   std::cout <<std::endl;
   return (output);
}
```

```
int main() {
    std::vector <int> vec1 {15, 2, 45, 12, 7};
    std::vector <int> vec2 {1};

    int size1= vec1.size();
    std::cout << "Output 1 : " << performOperation (vec1, size1) << std::endl;

    int size2 = vec2.size();
    std::cout << "Output 2 : " << performOperation (vec2, size2) << std::endl;

    /*
    for (int i=1;i<=vec2.size();++i) {
        std::cout << vec2 [i-1] << " " << i << "\t\t\t";
    }
    std::cout <<std::endl;*/
    return (0);
}

/*
Q6. Given an array of size N and you have to tell whether the array is perfect or not. An array is said
to be perfect if it’s reverse array matches the original array. If the array is perfect then print
"PERFECT" else print "NOT PERFECT".
Example 1:
Input : Arr[] = {1, 2, 3, 2, 1}
Output : PERFECT
Explanation:
Here we can see we have [1, 2, 3, 2, 1]
if we reverse it we can find [1, 2, 3, 2, 1]
which is the same as before.
So, the answer is PERFECT.
Example 2:
Input : Arr[] = {1, 2, 3, 4, 5}
Output : NOT PERFECT
*/

#include <iostream>
#include <vector>

void printOutput (std::vector <int> &);
void  performOperation (std::vector<int> &, int&);

void printOutput (std::vector <int> &received) {
    for (const auto& i: received) {
        std::cout <<  i  << " ";
    }
    std::cout << std::endl;
}


void  performOperation (std::vector<int> & received, int& vectorSize) {
    int i{0}, j{vectorSize-1};
    std::cout << "front " << received[i] << std::endl;
    std::cout << "back " << received[j] << std::endl;
    while (i<j) {
```

```
        //while (j>=i) {
            if (received[i++] == received [j--]) {
                std::cout << "Perfect \n";
            }
            else {
                std::cout << "Note perfect \n";
            }
            //--j;
        //}
        //++i;
    }
}
int main () {
    std::vector <int> vec1{};
    std::vector <int> vec2{1,2,3,4,5};

    //pushIntoVector(n);
    vec1.push_back(1);
    vec1.push_back(2);
    vec1.push_back(3);
    vec1.push_back(2);
      vec1.push_back(1);

    printOutput(vec1);
    int sizeofVector1=vec1.size();
    std::cout << sizeofVector1 << std::endl;
    std::cout << "==========================================================================
===\n";
    std::cout << "output 1: \n";
     performOperation (vec1,sizeofVector1);

     printOutput(vec2);
    int sizeofVector2=vec2.size();
    std::cout << sizeofVector2 << std::endl;
    std::cout << "==========================================================================
===\n";
    std::cout << "output 2: \n";
     performOperation (vec2,sizeofVector2);
     return (0);
}

/*
Q7. Given an array of length N, at each step it is reduced by 1 element. In the first step the maximum
element would be removed, while in the second step minimum element of the remaining array would
be removed, in the third step again the maximum and so on. Continue this till the array contains only 1
element. And find the final element remaining in the array.
Example 1:
Input:
N = 7
A[] = {7, 8, 3, 4, 2, 9, 5}
Ouput:
5
Explanation:
In first step ’9’ would be removed, in 2nd step
’2’ will be removed, in third step ’8’ will be
```

removed and so on. So the last remaining
element would be ’5’.
Example 2:
Input:
N = 8
A[] = {8, 1, 2, 9, 4, 3, 7, 5}
Ouput:
4
*/

```
#include <iostream>
#include <vector>
#include <algorithm>


void printOutput (std::vector <int> &);
int   performOperation (std::vector<int> &, int&);

void printOutput (std::vector <int> &received) {
    for (const auto& i: received) {
        std::cout <<  i  << " ";
    }
    std::cout << std::endl;
}

int performOperation (std::vector<int> &received, int& vectorSize) {
    std::sort(received.begin(), received.end());
    //int startPos = received.begin();
    //int endPos = received.end();
    printOutput(received);
    //int i{0}, j{vectorSize-1};
    //for (int i=vectorSize-1;i>0;--i) {
        //std::cout << received[i] << " \t" << std::endl;
        //if (received[])
        //std::cout << std::endl;
        //received[i+1] = received[i];

        //if (received[i] > received[i-1]) {
        //    received[i+1] = received [i];
        //}
    //}
    //for(const auto& i : received) {
    //    received.push
    //}
    int i{0};
    while(i <= vectorSize-2 ) {
        std::cout << "Enterd into if ==========================\n";
        received.pop_back();
        std::reverse(received.begin(), received.end());
        std::cout << "Pop & Rev done ========================== " << vectorSize << std::endl;
        printOutput(received);
        ++i;
    }

    printOutput(received);
```

```
}


int main () {
      std::vector <int> vec1{};
     std::vector <int> vec2{8,1,2,9,4,3,7,5};

     //pushIntoVector(n);
     vec1.push_back(7);
     vec1.push_back(8);
     vec1.push_back(3);
     vec1.push_back(4);
      vec1.push_back(2);
      vec1.push_back(9);
      vec1.push_back(5);

     printOutput(vec1);
     int sizeofVector1=vec1.size();
     std::cout << sizeofVector1 << std::endl;
     std::cout << "=========================================================================
===\n";
     std::cout << "output 1: \n";
      performOperation (vec1,sizeofVector1);

      printOutput(vec2);
     int sizeofVector2=vec2.size();
     std::cout << sizeofVector2 << std::endl;
     std::cout << "=========================================================================
===\n";
     std::cout << "output 2: \n";
      performOperation (vec2,sizeofVector2);
      return (0);
}

/*
Q8. Given an array of N distinct elements, the task is to find all elements in array except two greatest
elements in sorted order.
Example 1:
Input :
a[] = {2, 8, 7, 1, 5}
Output :
1 2 5
Explanation :
The output three elements have two or
more greater elements.
Example 2:
Input :
a[] = {7, -2, 3, 4, 9, -1}
Output :
-2 -1 3 4
*/


#include <iostream>
#include <vector>
```

```
#include <algorithm>


void printOutput (std::vector <int> &);
void   performOperation (std::vector<int>& , int&);

void printOutput (std::vector <int> &received) {
    for (const auto& i: received) {
        std::cout <<  i  << " ";
    }
    std::cout << std::endl;
}

void   performOperation (std::vector<int>&  received, int& vectorSize) {
    std::vector <int> resultVector(vectorSize-2); //except two greatest
    std::cout << "Size of new vector " << resultVector.size() << std::endl;
    std::sort (received.begin(), received.end());


    //while {
    //    received.pop_back();
    //}
    std::copy (received.begin(),received.end(),resultVector.begin());
    printOutput(resultVector);
    std::cout << std::endl;
    return;
}



int main () {
    std::vector <int> vec1{};
   std::vector <int> vec2{7,-2,3,4,9,-1};

    //pushIntoVector(n);
    vec1.push_back(2);
    vec1.push_back(8);
    vec1.push_back(7);
    vec1.push_back(1);
    vec1.push_back(5);

    printOutput(vec1);
    int sizeofVector1=vec1.size();
    std::cout << sizeofVector1 << std::endl;
    std::cout << "================================================================
==\n";
    std::cout << "output 1: \n";
    performOperation (vec1,sizeofVector1);

    printOutput(vec2);
    int sizeofVector2=vec2.size();
    std::cout << sizeofVector2 << std::endl;
    std::cout << "================================================================
==\n";
    std::cout << "output 2: \n";
```

```
    performOperation (vec2,sizeofVector2);
    return (0);
}

/*
Q9. Write a program to find the sum of the given series 1+2+3+ . . . . .(N terms)
Example 1:
Input:
N = 1
Output: 1
Explanation: For n = 1, sum will be 1.
Example 2:
Input:
N = 5
Output: 15
Explanation: For n = 5, sum will be 1
+ 2 + 3 + 4 + 5 = 15.
*/


#include <iostream>
//AP series N(N+1)/2
int getSumSeries(int &);

int getSumSeries(int & received) {
    std::cout << received << std::endl;
    //int product = received
    return (received*(received+1)/2);

}

int main() {
    int N{NULL};
    std::cout << "Enter value of n \n";
    std::cin >> N;

    if (N <= 0) {
        exit(-1);
    }
    std::cout << getSumSeries (N) << std::endl;
    return (0);
}

/*
Q10. Given a number N. Your task is to check whether it is fascinating or not.
Fascinating Number: When a number(should contain 3 digits or more) is multiplied by 2 and 3 ,and
when both these products are concatenated with the original number, then it results in all digits from 1
to 9 present exactly once.
Example 1:
Input:
N = 192
Output: Fascinating
Explanation: After multiplication with 2
and 3, and concatenating with original
number, number will become 192384576
```

which contains all digits from 1 to 9.
Example 2:
Input:
N = 853
Output: Not Fascinating
Explanation: It’s not a fascinating
number.
*/

```
#include <iostream>
#include <vector>
#include <algorithm>

void printOutput (std::vector <int> &);
void performOperation(int&);
void performCheck(std::vector<int> &);


void performCheck(std::vector<int>& received) {
    printOutput(received);
    for (int i=0;i<received.size();++i) {
        /*if (1 < i < 9) {
            std::cout << "Fascinating \n";
        } else {
            std::cout << "Not fascinating \n";
        }*/
        if (received[i] == received [i+1]) {
            std::cout << "Not fascinating \n";
            break;
        } else {
            std::cout << "Fascinating \n";
        }
    }
}

void printOutput (std::vector <int> &received) {
    for (const auto& i: received) {
        std::cout <<  i  << " ";
    }
    std::cout << std::endl;
}

void performOperation(int& received) {
   std::vector <int> resultVector {};
    int numberMultiply2 = (2*received);
    int numberMultiply3 = (3*received);
    std::cout << received << " "<< numberMultiply2 << " " << numberMultiply3 << std::endl;
    while (received != 0){
         int result = received %10;
         std::cout << "Enter into while " << result << std::endl;
         received /= 10;
         //resultVector.push_back(result);
        resultVector.insert(resultVector.begin(),result);
    }
```

```
    while (numberMultiply2 != 0){
        int result = numberMultiply2 %10;
        std::cout << "Enter into while " << result << std::endl;
        numberMultiply2 /= 10;
        //resultVector.push_back(result);
        resultVector.insert(resultVector.begin()+3,result);
    }
    while (numberMultiply3 != 0){
        int result = numberMultiply3 %10;
        std::cout << "Enter into while " << result << std::endl;
        numberMultiply3 /= 10;
        //resultVector.push_back(result);
        resultVector.insert(resultVector.begin()+6,result);
    }

    //resultVector.push_back(numberMultiply2);
    //resultVector.push_back(numberMultiply3);
    //performCheck(received);
    //performCheck(numberMultiply2);
    //performCheck(numberMultiply3);
    //std::reverse(resultVector.begin(),resultVector.end());
    printOutput(resultVector);
    performCheck(resultVector);
    //return (received);
}

int main() {
    int N{0};
    std::cout << "Enter n \n";
    std::cin >> N;
    performOperation (N);
    return(0);
}

/*
Bonus Question
Given an array of even size N, task is to find minimum value that can be added to an element so that array become balanced. An array is balanced if the sum of the left half of the array elements is equal to the sum of right half.
Example 1:
Input:
N = 4
arr[] = {1, 5, 3, 2}
Output: 1
Explanation:
Sum of first 2 elements is 1 + 5 = 6,
Sum of last 2 elements is 3 + 2 = 5,
To make the array balanced you can add 1.
Example 2:
Input:
N = 6
arr[] = { 1, 2, 1, 2, 1, 3 }
Output: 2
Explanation:
Sum of first 3 elements is 1 + 2 + 1 = 4,
```

```
Sum of last three elements is 2 + 1 + 3 = 6,
To make the array balanced you can add 2.
*/

#include <iostream>
#include <vector>
#include <algorithm>


void printOutput (std::vector <int> &);
int performOperation(std::vector <int>&,int&, int&);

void printOutput (std::vector <int> &received) {
    for (const auto& i: received) {
        std::cout <<  i  << " ";
    }
    std::cout << std::endl;
}

int performOperation(std::vector <int>& received, int& vectorSize, int& midPoint) {
    int temp1{0},temp2{0};
    for (int i=0;i<midPoint;++i) {
        temp1 += received[i];
    }
    for(int i=midPoint;i<vectorSize;++i) {
        temp2 += received[i];
    }
    //std::cout << temp1 << " " << temp2 << std::endl;

    int finalResult{0};
    if (temp1 > temp2) {
        finalResult = (temp1 - temp2);
    }
    else {
        finalResult = (temp2 - temp1);
    }
    return finalResult;

}


int main() {
    std::vector <int> vec1{};
 std::vector <int> vec2{1,2,1,2,1,3};

    //pushIntoVector(n);
    vec1.push_back(1);
    vec1.push_back(5);
    vec1.push_back(3);
    vec1.push_back(2);


    printOutput(vec1);
    int sizeofVector1=vec1.size();
    int midPoint1=((0+sizeofVector1)/2);
```

```
    std::cout << sizeofVector1 << " " << midPoint1<< std::endl;
    std::cout << "==========================================================================
==\n";
    std::cout << "output 1: " << performOperation (vec1,sizeofVector1, midPoint1) << std::endl;

    printOutput(vec2);
    int sizeofVector2=vec2.size();
    int midPoint2=((0+sizeofVector2)/2);
    std::cout << sizeofVector2 << std::endl;
    std::cout << "==========================================================================
==\n";
    std::cout << "output 2: " << performOperation (vec2,sizeofVector2, midPoint2) << std::endl;
    return (0);
}
```