using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace codechal190216

{

class Program

{

static void Main(string[] args)

{

int n = Convert.ToInt32(Console.ReadLine());

float div = 0;

div = 100 + (float)100 / n \* (n - 1);

Console.WriteLine("{0:0.00}",div);

Console.ReadLine();

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace codechal190216

{

class Program

{

public static double maxDistance(int n, int fuel)

{

double dist\_covered = 0;

while (n > 0)

{

dist\_covered += (double)fuel / n;

n -= 1;

}

return dist\_covered;

}

static void Main(string[] args)

{

int fuel = 100;

int n = int.Parse(Console.ReadLine());

double num=Program.maxDistance(n, fuel);

Console.WriteLine("{0:0.00}",Math.Round(num,2));

Console.ReadLine();

}

}

}

Page of

**Bikes**

**Bikes**

There are n bikes and each can cover 100 km when fully fueled. What is the maximum amount of distance you can go using n bikes? You may assume that all bikes are similar and a bike takes 1 litre to cover 1 km. [Hint: Fuel can be transferred from 1 bike to another]

**Input Format**

Input consists of a single integer that corresponds to n.

**Output Format**

Output consists of a float that corresponds to the maximum distance travelled. Output is displayed correct to 2 decimal places.

**Sample Input:**

2

**Sample Output:**

150.00

Page of

**Min Heap**

**Min Heap**

Write a program to find whether a given complete binary tree satisfies the min heap property.

**Complete Binary Tree:** A **complete binary tree** is a binary tree in which every level, *except possibly the last*, is completely filled, and all nodes are as far left as possible.

A **heap** is a specialized tree-based data structure that satisfies the *heap property:* If A is a parent node of B then the key of node A is ordered with respect to the key of node B with the same ordering applying across the heap. In a **min heap** the keys of parent nodes are always less than or equal to those of the children and the lowest key is in the root node.

**Input and Output Format:**

Input consists of n+1 integers where n corresponds to the number of elements in the tree.

The following n integers are the elements of the tree in level order. [i.e the 1st element is the root of the tree (Level 0), the 2nd and 3rd elements are in level 1 and so on...]

Output consists of a single string that is either yes or no. Output yes if the tree is a min heap.

**Sample Input:**

3

1

2

3

**Sample Output:**

Yes

Page of

**Chess Board 1**

**Chess Board 1**

Given the position of a bishop and a queen in a n\*n chessboard, mark the remaining positions in the chess board as follows:

'\*' --- if it is under attack from bishop

'$' --- if it is under attack from queen

'%' --- if it is under attack from both queen and bishop.

'.' --- if it is not under attack.

**Input and Output Format:**

Input consists of 5 integers where first integer, n, corresponds to the size of the chess board.

Second and third integers correspond to the x and y coordinates of the bishop respectively, and fourth and fifth integers correspond to the x and y coordinates of the queen respectively.

Output consists of a *nxn* matrix obtained by applying the above rules.

**Sample Input:**

4

2

2

3

4

**Sample Output:**

\*$\*$

.B$$

%$%Q

..$%

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace codechal190216

{

class Program

{

static void Main(string[] args)

{

int n = Convert.ToInt32(Console.ReadLine());

int[] arr = new int[n];

for (int i = 0; i < n; i++)

{

arr[i] = Convert.ToInt32(Console.ReadLine());

}

int flag = 0;

for (int i = 0; i < n; i++)

{

int c1 = (2 \* i) + 1;

int c2 = (2 \* i) + 2;

if (c1 < n && c2 < n)

{

if (arr[i] > arr[c1] || arr[i] > arr[c2])

{

flag = 1;

break;

}

}

else if (c1 < n)

{

if (arr[i] > arr[c1])

{

flag = 1;

break;

}

}

}

if (flag == 0)

Console.WriteLine("yes");

else

Console.WriteLine("no");

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace codechal190216

{

class Program

{

static void Main(string[] args)

{

int n = Convert.ToInt32(Console.ReadLine());

int bx = Convert.ToInt32(Console.ReadLine());

int by = Convert.ToInt32(Console.ReadLine());

int qx = Convert.ToInt32(Console.ReadLine());

int qy = Convert.ToInt32(Console.ReadLine());

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

{

for (int j = 1; j <= n; j++)

{

if (i == bx && j == by)

Console.Write("B");

else if (i == qx && j == qy)

Console.Write("Q");

else if (Math.Abs(bx - i) == Math.Abs(by - j))

{

if ((qx == i || qy == j) || (Math.Abs(qx - i) == Math.Abs(qy - j)))

{

Console.Write("%");

}

else

{

Console.Write("\*");

}

}

else if ((qx == i || qy == j) || (Math.Abs(qx - i) == Math.Abs(qy - j)))

{

Console.Write("$");

}

else

{

Console.Write(".");

}

}

Console.WriteLine();

}

Console.ReadLine();

}

}

}