using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace CODECHAL030216

{

class Program

{

static void Main(string[] args)

{

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

int x1=Convert.ToInt32(Console.ReadLine());

int y1=Convert.ToInt32(Console.ReadLine());

int x2=Convert.ToInt32(Console.ReadLine());

int y2=Convert.ToInt32(Console.ReadLine());

char[,] symbol = new char[n, n];

int a1, a2;

int i, j;

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

{

for (j = 0; j < n; ++j)

{

a1 = (x1 - 1) - i;

a2 = (y1 - 1) - j;

if ((i == x1 - 1) && (j == y1 - 1))

{

symbol[i, j] = 'B';

}

else if ((i == x2 - 1) && (j == y2 - 1))

{

symbol[i, j] = 'R';

}

else if (i == x2 - 1 || j == y2 - 1)

{

if (Math.Abs(a1) == Math.Abs(a2))

{

symbol[i, j] = '%';

}

else

{

symbol[i, j] = '$';

}

}

else if (Math.Abs(a1) == Math.Abs(a2))

{

symbol[i, j] = '\*';

}

else

symbol[i, j] = '.';

}

}

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

{

for (j = 0; j < n; ++j)

{

Console.Write(symbol[i, j]);

}

Console.WriteLine();

}

Console.ReadLine();

}

}

}

Page of

**Chess Board 2**

**Chess Board 2**

Given the position of a bishop and a rook 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 rook

'%' --- if it is under attack from both rook 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 rook respectively.

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

**Sample Input:**

4

2

2

3

3

**Sample Output:**

\*.%.

.B$.

%&R$

..$\*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace ConsoleApplication15

{

    class Program

    {

        static void Main(string[] args)

        {

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

            int[] a = new int[n];

            int flag=0,l=0,r=0;

            for(int i=0;i<n;i++)

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

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

            {

                l=(2\*i)+1;

                r=(2\*i)+2;

                if(l<n && r<n)

                {

                if(a[i]>a[l] && a[i]<a[r])

                    continue;

                else

                {

                    flag=1;

                    break;

                }

                }

                else if (l < n)

                {

                    if (a[i] > a[l])

                        continue;

                    else

                    {

                        flag = 1;

                        break;

                    }

                }

            }

            if (flag == 1)

                Console.WriteLine("no");

            else

                Console.WriteLine("yes");

            Console.ReadLine();

        }

    }

}

​

Page of

**BST**

**BST**

Write a program to find whether a given complete binary tree is a Binary Search Tree or not.

**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 **binary search tree** (**BST**), sometimes also called an **ordered** or **sorted binary tree**, is a node-based binary tree data structure where each node has a comparable key (and an associated value) and satisfies the restriction that the key in any node is larger than the keys in all nodes in that node's left subtree and smaller than the keys in all nodes in that node's right sub-tree.

**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 BST.

**Sample Input:**

3

3

2

4

**Sample Output:**

yes