

**УНИВЕРСИТЕТ ПО БИБЛИОТЕКОЗНАНИЕ И ИНФОРМАЦИОННИ ТЕХНОЛОГИИ**

**ФАКУЛТЕТ „ИНФОРМАЦИОННИ НАУКИ“**

**ПРОФЕСИОНАЛНО НАПРАВЛЕНИЕ „ИНФОРМАТИКА И КОМПЮТЪРНИ НАУКИ“**

**СПЕЦИАЛНОСТ „ИНФОРМАЦИОННИ СИСТЕМИ И ТЕХНОЛОГИИ“**

**КУРСОВА РАБОТА**

**ПО УЧЕБНА ДИСЦИПЛИНА**

**ОБЕКТНО ОРИЕНТИРАНО ПРОГРАМИРАНЕ**

**Изготвил: Научен ръководител:** проф. И. Иванов

Кристин Даниел Димитрова

Ф.№ 533-ИМЗ

София

2018

Съдържание

[Задача 1 – Коледна елха 2](#_Toc503265047)

[Условие 2](#_Toc503265048)

[Решение 2](#_Toc503265049)

[Стъпка 1 2](#_Toc503265050)

[Стъпка 2 2](#_Toc503265051)

[Стъпка 3 3](#_Toc503265052)

[Стъпка 4 4](#_Toc503265053)

[Тест 5](#_Toc503265054)

[Задача 2 – Human, Student, Worker 5](#_Toc503265055)

[Условие 6](#_Toc503265056)

[Решение 6](#_Toc503265057)

[Стъпка 1 6](#_Toc503265058)

[Стъпка 2 6](#_Toc503265059)

[Стъпка 3 7](#_Toc503265060)

[Стъпка 4 7](#_Toc503265061)

[Стъпка 5 8](#_Toc503265062)

[Тест 8](#_Toc503265063)

[Задача 3 – Умната Лили 9](#_Toc503265064)

[Условие 10](#_Toc503265065)

[Вход 10](#_Toc503265066)

[Изход 10](#_Toc503265067)

[Примерен вход и изход 10](#_Toc503265068)

[Решение 10](#_Toc503265069)

[Стъпка 1 11](#_Toc503265070)

[Стъпка 2 11](#_Toc503265071)

[Стъпка 3 11](#_Toc503265072)

[Стъпка 4 11](#_Toc503265073)

[Стъпка 5 12](#_Toc503265074)

[Тест 13](#_Toc503265075)

# Задача 1 – Коледна елха

## Условие

Напишете програма, която въвежда число n (1 ≤ n ≤ 100) и печата **коледна елха** с размер n като в примерите по-долу:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **вход** | **изход** |  | **вход** | **изход** |  | **вход** | **изход** |  | **вход** | **изход** |
| 1 | |  \* | \* | 2 | |  \* | \*  \*\* | \*\* | 3 | |  \* | \*  \*\* | \*\*  \*\*\* | \*\*\* | 4 | |  \* | \*  \*\* | \*\*  \*\*\* | \*\*\*  \*\*\*\* | \*\*\*\* |

## Решение

### Стъпка 1

На първи етап трябва да въведем число n, което да бъде базата на нашата програма.

Правим това чрез:  
  
int n = int.Parse(Console.ReadLine());

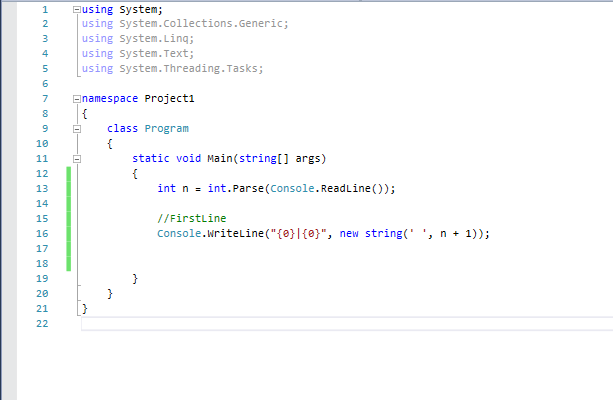
### Стъпка 2

Гледайки първи ред на дадените примери, установяваме, че той се състои от равен брой интервали от двете страни (n+1), разделени с права черта (|).

За да изобразим първия ред от нашата фигура използваме:

Console.WriteLine("{0}|{0}", new string(' ', n + 1));

, където чрез плейсхолдъри добавяме нов стринг от идентични n+1 брой интервали, между които при всички случаи има една единствена права черта.



С това първи ред от нашата програма е готов.

### Стъпка 3

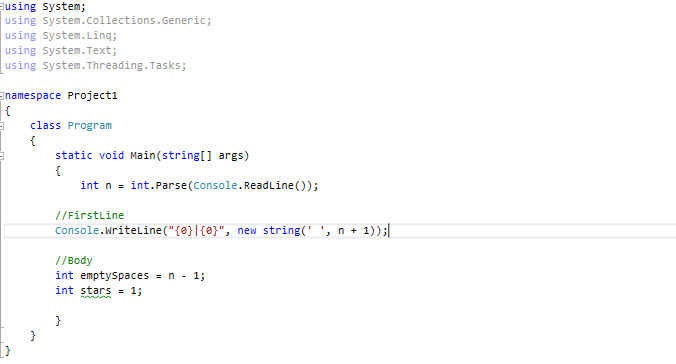
Започваме да „строим“ тялото на нашата елха.

Забелязваме, че лявата и дясната част на елхата са аналогични, разделени с права черта (|). Ще започнем с лявата част, а после ще добавим дясната. За първи ред всяка част се състои от два идентични (n-1) брой интервали, разделени със звездичка (\*). С преминаването на всеки следващ ред броят на звездичките се увеличава, докато не достигне n на брой, за сметка на броя на интервалите от лявата страна.

Правим си променливи за броя на интервалите и за броя на звездичките.

int emptySpaces = n - 1;

int stars = 1;



### Стъпка 4

Забелязали сме, че без да броим първи ред на нашата програма, или върха на нашата елха, който вече сме създали в стъпка 2, дръвчето се състои от точно n на брой редове.

Създаваме for цикъл, който започва от 0 и стига до n-1, в който „построяваме“ всеки ред на части.

Започваме с n-1 брой интервали,намаляващи с всяка итерация на цикъла

Console.Write(new string(' ', emptySpaces));

emptySpaces--;

, последвани от 1 звездичка.

Console.Write(new string('\*', stars));

Следва константа, състояща се от интервал, права черта, интервал

Console.Write(" | ");

,последвана от същия брой звездички(като за начало - 1) за дясната страна на дървото.

Console.Write(new string('\*', stars));

Броят на звездичките се увеличава с всяка итерация и преминаваме на следващия ред от дървото.

stars++;

Console.WriteLine();

Целият ни цикъл изглежда ето така:

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

{

Console.Write(new string(' ', emptySpaces));

emptySpaces--;

Console.Write(new string('\*', stars));

Console.Write(" | ");

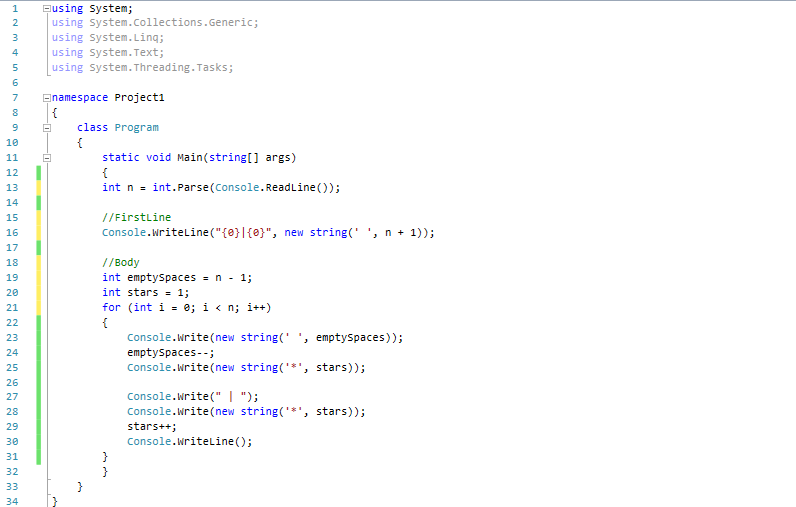
Console.Write(new string('\*', stars));

stars++;

Console.WriteLine();

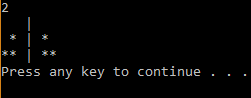
}

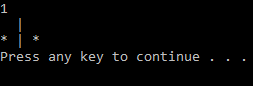
С това нашата програма е завършена. В цялостен вид тя изглежда ето така:

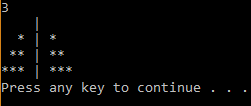
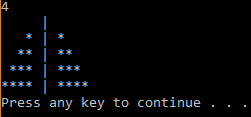


## Тест

Ще тестваме нашето решение със стойностите на n, дадени в условието.







При всички тестове се получава желаният резултат. Задачата е решена.

# Задача 2 – Human, Student, Worker

## Условие

Дефинирайте клас Human със свойства Име и Фамилия. Дефинирайте класове Student и Worker – наследници на класа Human. Нека класът Student да има свойство Факултетен номер и метод, който да изчислява средния успех на ученика при въведен от потребителя брой и вид оценки. Нека класът Worker да има свойство Компания, чийто служител е работникът, и метод, изчисляващ заплатата на работника за един час при въведени от потребителя месечна заплата\* и часове на работа. Тествайте програмата като създадете по 2 обекта от всеки клас и им приложите съответния метод.  
\**Приема се, че в един месец има 4 работни седмици.*

## Решение

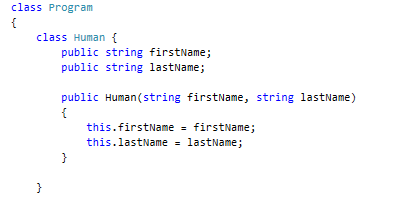
### Стъпка 1

Създаваме базовия клас Human, в който определяме променливи за име и фамилия .

public string firstName;

public string lastName;

В класа създаваме конструктор Human, който ще използваме, за да създадем обекти от дадения клас. Правим това чрез ключовата дума this.



### Стъпка 2

След като вече сме си създали parent-класа Human, е време да създадем първия му клас наследник – Student. В него добавяме допълнителното свойство Факултетен номер, характерно само за класа Student, но не и за базовия клас Human.

public string fNumber;

Следователно, трябва да направим нов конструктор за класа Student, който да съдържа свойствата, наследени от базовия клас, но да бъде включено и новото поле.

public Student(string firstName,string lastName, string fNumber) :base(firstName,lastName) {

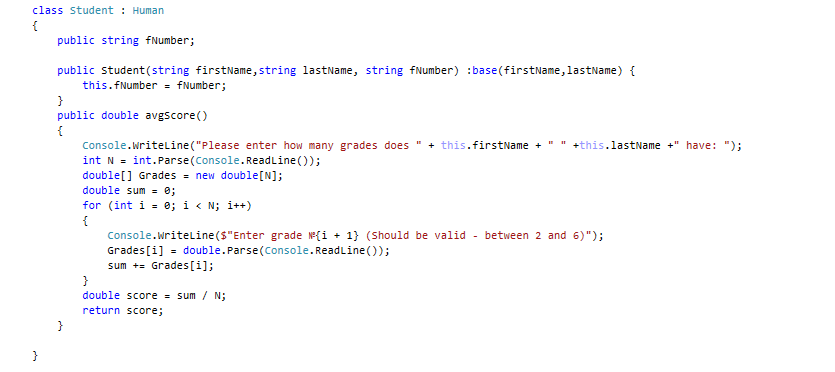
this.fNumber = fNumber;

}

### Стъпка 3

Знаем, че класът Student трябва да притежава метод, който да изчислява средния успех на ученика при въведен от потребителя брой и вид оценки. Наименуваме го avgScore(). На първи етап изискваме от потребителя да въведе колко оценки ще има нашия ученик. Съхраняваме въведената бройка в променливата N. Създаваме масив от N на брой числа, в който ще съхраняваме оценките, които потребителя ще въведе. Избираме масив от тип double, защото оценките могат да не бъдат цели числа. Правим променлива sum, която за начало е 0, и в която ще “натрупваме” сбора от оценките. Време е да поискаме от потребителя да “оцени” нашия ученик, като за целта използваме цикъл, чрез който “напълваме” масива от оценки и присвояваме на sum стойността на техния сбор. Разделяме сбора на броя на оценките и получаваме средния успех на ученика.

С това класът Student е готов.



### Стъпка 4

Време е да създадем втория клас наследник на Human – класа Worker. Знаем, че той наследява името и фамилията от класа-родител, но има и допълнително свойство за компанията, в която работи дадения човек. Създаваме си променливата company.

public string company;

След това аналогично на другия клас наследник, трябва да създадем нов конструктор, наследяващ свойствата на базовия клас, включващ и новата променлива company.

public Worker(string firstName,string lastName, string company) :base(firstName,lastName)

{

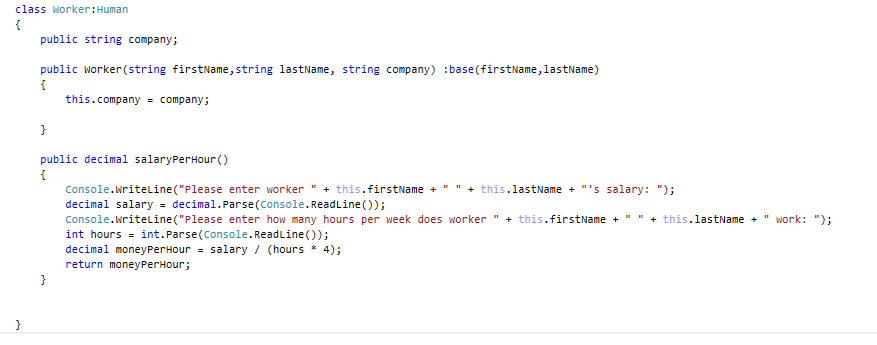
this.company = company;

}

### Стъпка 5

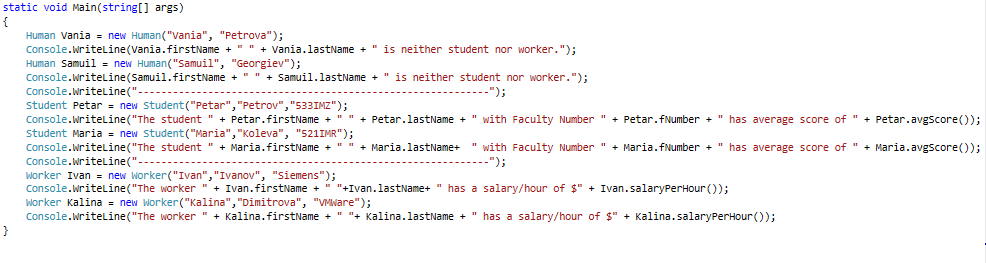
Класът Worker трябва да притежава метод, който при въведена месечна заплата и отработени часове изчислява почасовата заплата на работника. Като за начало изискваме от потребителя да въведе заплатата и я съхраняваме в променлива salary от тип decimal. Следва да разберем колко часа седмично работи нашия работник. Съхраняваме тези данни в променливата от тип integer - hours. Почасовата заплата получаваме като разделим месечната такава на броя часове на седмица, умножен по 4 седмици. Съхраняваме я в променлива moneyPerHour.

Вече готовият ни клас изглежда ето така:

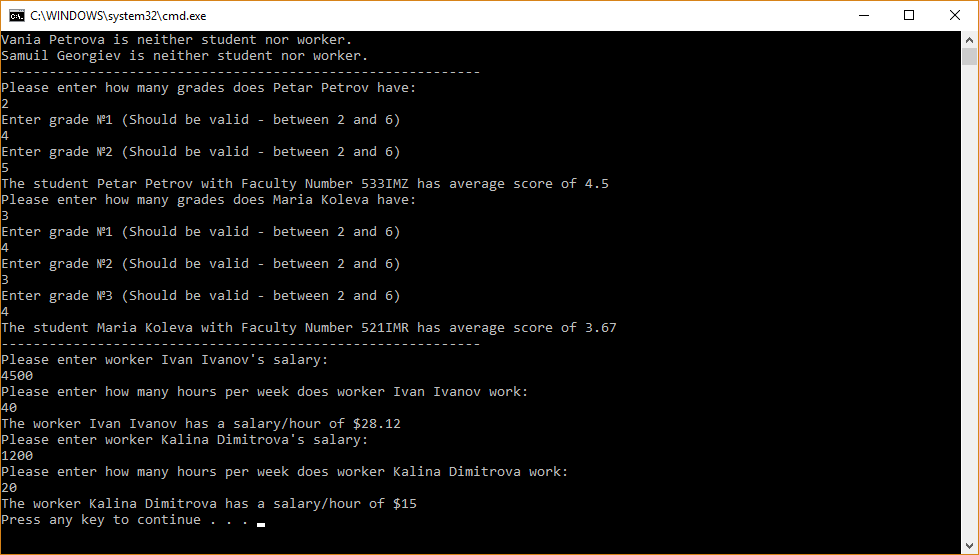


## Тест

След като сме изпълнили всички изисквания от задачата, свързани с класовете, трябва да създадем по 2 екземпляра от всеки клас, с които да тестваме нашата програма. Правим това в Main метода.



Резултат:



Функционалността работи, задачата е решена.

# Задача 3 – Умната Лили

## Условие

Лили вече е на **N години**. За всеки свой **рожден ден** тя получава подарък. За **нечетните** рождени дни (**1, 3, 5...n**) получава **играчки**, а за всеки **четен** (**2, 4, 6...n**) получава **пари**. За **втория рожден ден** получава **10.00 лв**, като **сумата се увеличава с 10.00 лв, за всеки следващ четен рожден ден** (**2 -> 10**, **4 -> 20**, **6 -> 30**...и т.н.). През годините Лили тайно е спестявала парите. **Братът** на Лили, **в годините**, които тя **получава пари**, **взима по 1.00 лев** от тях. Лили **продала играчките** получени през годините, **всяка за P лева** и **добавила** сумата **към спестените пари**. С парите искала да си **купи пералня за X лева**. Напишете програма, която да пресмята, **колко пари е събрала** и дали ѝ **стигат да си купи пералня**.

### Вход

От конзолата се прочитат **3 числа**, на отделни редове:

* **Възраста** на Лили – **цяло число** в интервала **[1...77]**
* **Цената на пералнята** – число в интервала **[1.00...10 000.00]**
* **Единична цена** **на играчка** – **цяло** **число** в интервала **[0...40]**

### Изход

Да се отпечата на конзолата един ред:

* Ако парите на Лили са достатъчни:
  + **“**Yes! {N}**”** – където **N** е остатъка пари след покупката
* Ако парите не са достатъчни:
  + **“No! {М}“** – където **M** е сумата, която не достига
* Числата **N** и **M** трябва да за **форматирани до вторият знак след десетичната запетая**.

### Примерен вход и изход

|  |  |  |
| --- | --- | --- |
| **вход** | **изход** | **Коментари** |
| 10  170.00  6 | Yes! 5.00 | **Първи рожден ден** получава **играчка**; **2ри -> 10лв**; 3ти -> играчка;  **4ти** -> 10 + 10 = **20лв**; **5ти** -> играчка; **6ти** -> 20 + 10 = **30лв**; **7ми** -> играчка;  **8ми** -> 30 + 10 = **40лв**; **9ти** -> играчка; **10ти** -> 40 + 10 = **50лв**.  **Спестила е** -> 10 + 20 + 30 + 40 + 50 = **150лв**. Продала е **5 играчки по 6 лв** = **30лв**.  **Брат ѝ взел 5 пъти по 1 лев** = **5лв**. **Остават** -> 150 + 30 – 5 = **175лв**.  **175 >= 170** (цената на пералнята) **успяла** е да я купи и са и **останали** 175-170 = **5 лв.** |
| 21  1570.98  3 | No! 997.98 | **Спестила е 550лв**. **Продала** е **11** играчки по **3 лв** = **33лв**. Брат ѝ **взимал 10** години по **1 лев** = **10лв**. **Останали** 550 + 33 – 10 = **573лв**  **573 < 1570.98** – **не е успяла** да купи пералня. **Не и достигат** 1570.98–573 = **997.98лв** |

## Решение

### Стъпка 1

На първи етап изискваме от потребителя да въведе годините на Лили, които съхраняваме в променлива от тип integer на име age. След това по условие ни трябва цената на пералнята, която съхраняваме в променлива от тип double на име price. Последният вход от потребителя е цената на една играчка, която по условие е цяло число. Съхраняваме я в променлива от тип integer – toyPrice.

Console.WriteLine("Please, enter Lily's age: ");

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

Console.WriteLine("Please, enter the price of the laundry machine: ");

decimal price = decimal.Parse(Console.ReadLine());

Console.WriteLine("Please, enter the price of one toy: ");

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

### Стъпка 2

Създаваме си целочислените променливи – toys, money, stolenMoney, които съответно представляват играчките, които Лили е получила за нечетните си рождени дни, парите, които е получила за четните си рождени дни и натрупаната сума от по 1лв, които братът и си е взимал. Като за начало на всичките променливи декларираме стойност 0.

int toys = 0;

int money = 0;

int stolenMoney = 0;

### Стъпка 3

Чрез for цикъл представяме всичките рождени дни на Лили до текущата и възраст, заедно със съответните подаръци, получени от нея. По условие знаем, че за четните си рождени дни момичето получава пари, които за втория и рожден ден са 10лв и за всеки следващ се увеличават с още 10лв. Също така парите, които брат и взима се увеличават с по 1лв на всеки неин четен рожден ден. Съответно натрупваме новите стойности в променливите money и stolenMoney. Ако рождения ден е нечетен, увеличаваме стойността на toys с 1.

Целият цикъл изглежда ето така:

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

{

if (i % 2 == 0)

{

money += i \* 10 / 2;

stolenMoney++;

}

else

{

toys++;

}

}

### Стъпка 4

Знаем, че Лили продава играчките си на цената, определена от потребителя и съхранена в променливата toyPrice. Сумата, която е спечелила от продажбите получаваме като умножим броя на играчките по цената им. Съхраняваме я в променливата soldToysMoney.

decimal soldToysMoney = toys \* toyPrice;

За да сметнем крайните пари на Лили трябва да си направим нова променлива totalMoney, в която да съберем парите, получени за рождените дни, плюс парите, спечелени от продадени играчки, и от нея да извадим парите, които брат и си е взимал през годините.

decimal totalMoney = soldToysMoney + money - stolenMoney;

### Стъпка 5

Време е да проверим дали Лили ще има достатъчно пари да си купи пералня. За да разберем, сравняваме цената, която потребителя е определил за пералнята, с парите, които Лили е събрала. Проверяваме дали сумата е по-голяма или равна на цената. Ако да – Лили ще има нова пералня и даже може да и останат спестявания на стойност – (totalMoney - price) . Ако не – тя ще трябва да събере още пари на стойност – (price – totalMoney).

if (totalMoney >= price)

{

Console.WriteLine("Yes! {0:f2}", totalMoney - price);

}

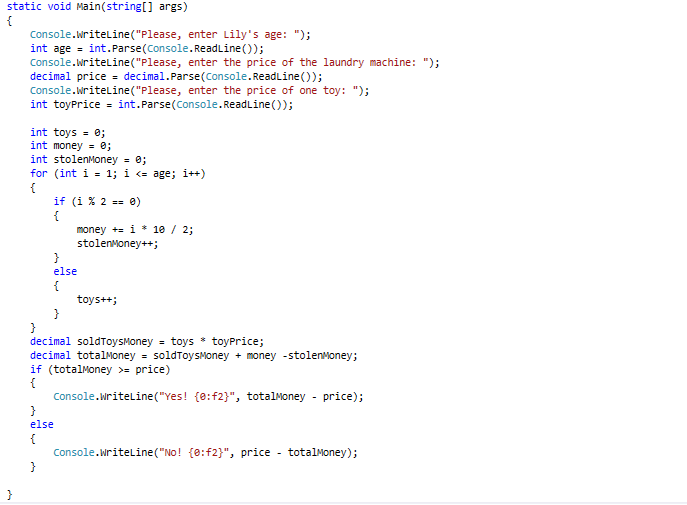
else

{

Console.WriteLine("No! {0:f2}", price - totalMoney);

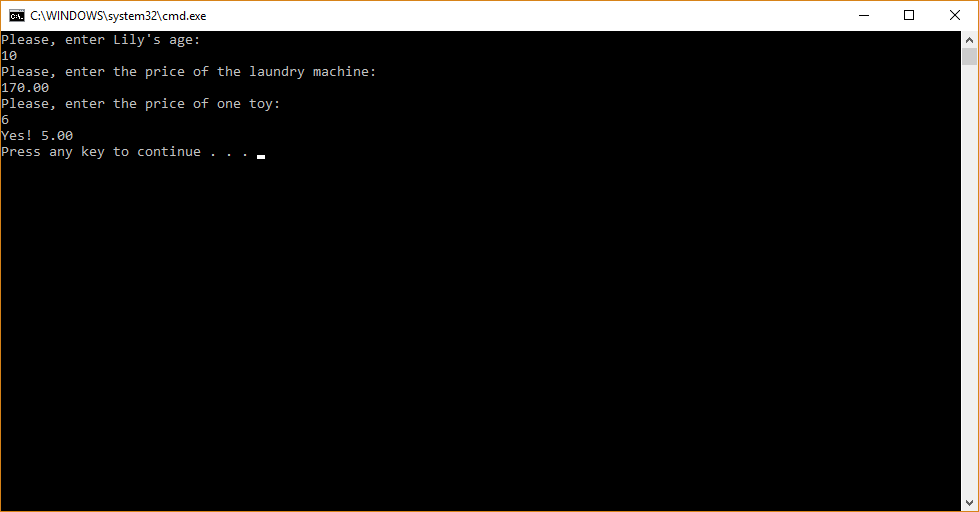
}

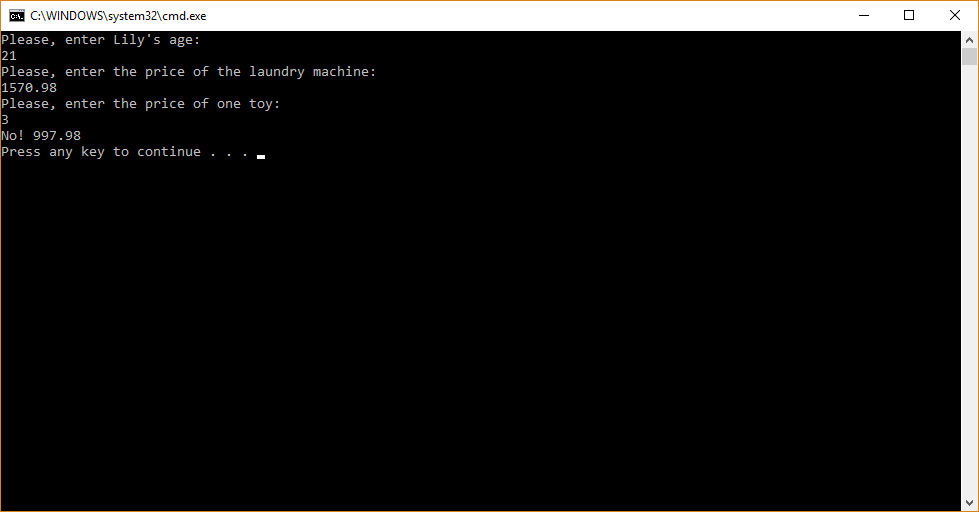
В цялостен вид програмата изглежда така:



### Тест

За да тестваме нашата програма, ще използваме примерните входове, дадени ни в условието.





Изглежда, че нашата програма работи правилно.