HÁSKÓLINN Í REYKJAVÍK
REYKJAVIK UNIVERSITY

Forritun
Programming

T-111-PROG

Lokapróf / Final exam

Kennari:	**Daníel B. Sigurgeirsson**
Dagsetning:	**14. nóvember 2016**
Tími:	**14:00 - 18:00**
Hjálpargögn:	**CodeBlocks þýðandi, formúlublað**

Nafn: _____

Kt.: _____

Prófið er 6 blaðsíður.

The exam is 6 pages

Gangi ykkur vel!

Good luck!

Í dæmunum er inntak frá notanda haft feitletrað. / In the examples, the input from the user is in boldface.

Ef þið eigið erfitt með að finna sérstafi á lyklaborðinu, þá eru nokkrir þeirra hér:
`{}[]()&*/\><=!"'%`

1. Minnsta talan (10% - 18 mínútur)

Skrifið forrit sem býður notandanum að slá inn N heiltölur (þar sem N er tala á bilinu 3 - 20 sem notandinn getur valið), athugið að tölurnar eru ekki slegnar inn í neinni sérstakri röð. Forritið prentar svo út **minnstu** töluna af þeim sem notandinn sló inn. Ef notandi velur töluna N utan við löglegt svið skal forritið hætta keyrslu.

Í forritinu skulu vera tvö föll: receiveInput() og findSmallest() (sjá neðar)

*Write a program which allows the user to enter N integers (where N is a number between 3 and 20 inclusively which the user picks), note that the numbers will not be entered in any particular order. The program then prints the **smallest** number of those entered by the user. If the user picks N outside of the legal range, the program quits.*

The program should use two functions: receiveInput() and findSmallest() (see below).

```
// This function returns true if the user entered a valid N value,
// but false otherwise.
bool receiveInput(int numbers[], int count);

// This function returns the smallest value found in the array.
// It assumes that the array is not empty, i.e. count >= 3.
int findSmallest(int numbers[], int count);
```

Dæmi / *Example*:

```
How many numbers will you enter?

5

3 2 1 4 5

The smallest number is 1
```

2. Rectangle (20% - 36 mínútur)

Skrifið klasann Rectangle. Klasinn er með smið sem tekur inn lengd og breidd ferhyrningsins. Í klasanum er fall sem skilar flatarmáli ferhyrningsins, og annað fall sem skilar ummáli hans. Operator << er yfirskrifaður, sem og samanburðarvirkinn operator ==. Tveir ferhyrningar teljast jafnir ef flatarmál þeirra er það sama.

Create the class Rectangle. The class has a constructor which accepts the width and height of a rectangle. The class has a method which returns the area of the rectangle, and a second method which returns the circumference. Operator << is overridden, as well as the equality comparison operator (i.e. operator ==). Two rectangles should be considered equal if their area is the same.

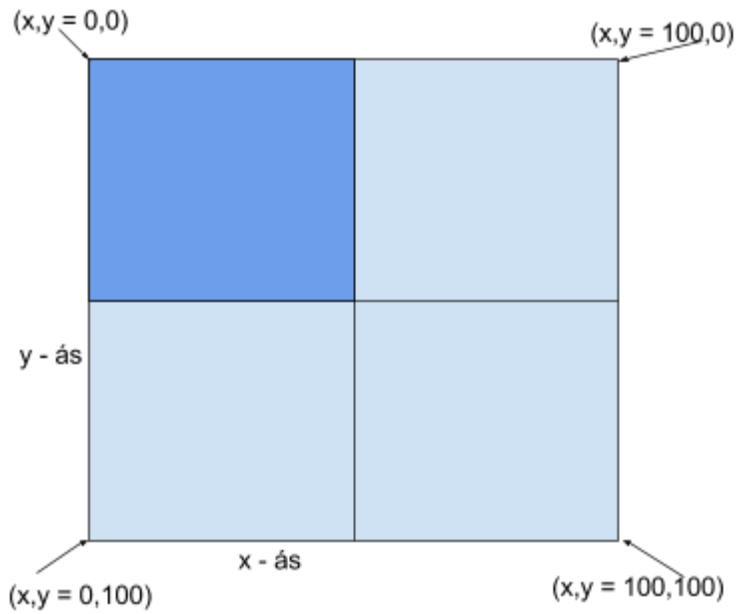
```
int main()
{
	Rectangle r(10,20);
	cout << r << endl;
	cout << r.area() << endl;
	cout << r.circumference() << endl;
	Rectangle r2(10,21);
	cout << r2 << endl;
	if (r == r2)
	{
		cout << "The two rectangles are equally large";
	}
	else
	{
		cout << "The two rectangles are not equal";
	}
}

// Expected output:

Width: 10, Height: 20
200
60
Width: 10, Height: 21
The two rectangles are not equal
```

3. Punktar (30% - 54 mínútur)

Skrifið forrit sem les inn upplýsingar um ótilgreindan fjölda punkta úr skrá og geymir þá í einum vector<>. Sérhver punktur heldur utan um x og y hnit, þar sem bæði x og y geta verið á bilinu 0 - 100. Forritið prentar svo út lista yfir þá punkta sem eru inni í ferhyrningi sem hylur "norðvestur" hluta svæðisins, þ.e. frá 0,0 upp í 50,50 (sjá skýringarmynd, athugið að myndin er eingöngu til útskýringar fyrir okkur en forritið veit ekkert af henni). Gera má ráð fyrir að hnitin í skránni séu öll lögleg. Nota skal klasa til að halda utan um sérhvert tilvik af punkti, og lógíkin sem segir til um hvort punkturinn sé í efri vinstri fjórðunginum eða ekki skal vera útfærð í sérstöku falli (sjá neðar).

Write a program which reads information about N points from a file and stores them in a single vector<>. Each point has x and y coordinates, where both x and y can be in the range 0 - 100. The program then prints a list of all the points which are within the upper left rectangle of the area, i.e. which are within the rectangle defined by the points 0,0 and 50,50 (see diagram, the diagram is only to help us visualize the problem, the program itself doesn't know about it). You may assume that all coordinates in the file are valid. You should use a class to manage the points, and the logic which determines if a point is in the upper left quadrant or not should be in a separate function.

```
bool isPointInUpperLeftQuadrant(Point p);
```



If the contents of the file Points.txt is as follows:

```
3 27
21 51
55 43
68 99
49 49
```

Then the expected output is:

```
Points in the upper left quadrant are:
3 27
49 49
```

4. Order (40% - 72 mínútur)

Skrifið klasann Order, sem heldur utan um pöntun í verslun. Sérhver pöntun getur innihaldið margar pöntunarlínur, þar sem sérhver lína segir til um nafn vöru, verð, fjölda, og afslátt sem við veitum viðskiptavininum. Til að halda utan um sérhverja línu þarf að útbúa annan klasa sem heitir OrderLine. Klasinn Order skal yfirskrifa << virkjann og prenta þar út pöntunina (sjá dæmi). OrderLine skal einnig yfirskrifa << virkjann, og prenta út eina línu í pöntun. Þá skal klasinn Order vera með aðgerð sem reiknar út og skilar heildarverði pöntunar. Neðantalið main() fall er gefið sem notar þessa klasa, og einnig er gefið úttak forritsins. Klasinn Order ætti að vera í sér .h/.cpp skrám, og klasinn OrderLine í sínum eigin .h/.cpp skrám.

Hint við útfærslu: þó svo að main() noti ekki OrderLine, þá skulið þið búa þann klasa til, og láta addLine() fallið búa til tilvik af honum og setja í vector sem klasinn Order á (þ.e. Order klasinn á meðlimabreytu af taginu vector<OrderLine>). Fallið getTotal() skoðar gögnin í þeim vector og notar þau til að skila upplýsingum um heildarverð í gegnum tilvísunarfæribreytur. Þetta fall væri jafnframt notað inni í operator << til að vita hvaða tölur eigi að prenta út neðst á pöntuninni. Við útfærslu á útprentun má gera ráð fyrir að allir dálkar séu af fastri breidd, og það að láta dálkana vera jafna er ekki aðalatriðið í þessu dæmi.

Write the class Order, which manages an order in a shop. Each order can contain multiple order lines, where each line specifies the name of a product, price, quantity bought, and a possible discount given to the customer. In order to store each line, you need to create a second class called OrderLine. The class Order should override the << operator and print the order (see example). The class OrderLine should also override the << operator, that implementation should print the given line in an order. Also, the class Order should contain a function which calculates and returns the total amount of the order. The main() function given below uses those classes, and the expected output is given as well. The class Order should be located in a separate .h/.cpp files, and the class OrderLine in its own .h/.cpp files.

Implementation hint: even though the main() method doesn't use the class OrderLine directly, you should create it nonetheless, and ensure that the addLine() method creates an instance of it and stores it in a vector belonging to the Order class. The function getTotal() should examine the data in this vector, and use it to return information about the total amount via reference parameters. You would also use this function inside the operator << for the class Order, such that it prints the correct total for the order. When implementing the printout, you may assume that all columns are of a fixed width, and the alignment of the columns is not the main priority in this assignment.

// See next page

```
int main()
{
	Order order;

	// The customer is buying 12 eggs at 60 kr. each.
	// There is no discount on eggs (in this case).
	order.addLine("eggs", 60, 12, 0);

	// The customer buys a single bread, the price of the bread
	// is 200 but we give the customer a 10% discount.
	order.addLine("bread", 200, 1, 10);

	// The customer buys 2 cartons of milk, the price of each carton is
	// 120 kr., and we then give the customer a 5% discount.
	order.addLine("milk", 120, 2, 5);

	int total = 0;
	int totalExcludingDiscount = 0;
	int discount = 0;
	order.getTotal(total, totalExcludingDiscount, discount);
	cout << "The total price is: " << total << endl;


	cout << order;


	return 0;
}


// The output from this function should be the following:

The total price is: 1128

Product    Price    Quantity   Discount    Total
================================================
eggs       60       12         0           720
bread      200      1          10          180
milk       120      2          5           228
================================================
Total excluding discount:                   1160
Total discount:                             32
Total:                                      1128
```