# Afbeeldingsresultaat voor pizza restaurantExam: Pizza time

When a lot of exams have to be reviewed, it is possible that the lectures do not have enough time to make a decent meal. Luckily there is always the possibility to order some pizza. The pizza restaurant has heard that we are lecturer from HBO-ICT and has asked us to make a system for printing digital receipts. It is now your task to develop a system for printing receipts for a pizza order.

The orders that can be done at the restaurant are luckily quite simple. It is only possible to order two types of products, namely: softdrinks or pizza. For softdrinks the employee has to enter the price per can and the number of cans to calculate the total price. For pizzas calculating the price is a bit more difficult. Because a lot of different pizzas can be ordered, together with the restaurant it has been decided the employee will fill out the list of ingredients and price per ingredient for every pizza. From the list of ingredients the item price of the pizza can now be calculated.

The system does not have to store a database of all possible softdrinks and pizzas, nor does it have to store a database of all possible ingredients. The employee will fill out per pizza which ingredients are on the pizza. Printing of the receipt can be done by writing to std::cout.

**Please note**: You don’t have to create a menu or other user interface for the system. It’s sufficient to add some softdrinks and pizzas to the order class in your main function (please check the comments in main.cpp for information on which calls you must add).

## Example output

Output of the requested application could look like this:

Welcome to Cheesy Pizzas, you have ordered the following products:

3x Soft-Drink Coca-Cola, price: 2.15

1x Soft-Drink Orange-Juice, price: 2.65

2x Pizza Margherita, price: 6.75

Ingredients: Dough (3) Tomato (2) Mozzarella (1) Oregano (0.75)

2x Pizza Salami, price: 7.75

Ingredients: Dough (3) Tomato (2) Paprika (1) Salami (1.75)

Total price of your order: 38.1

## The steps to build the system

* Design the system on a piece of paper. Think of all the classes and their attributes and methods. Draw a class diagram of the best solution. You don’t have to turn in the design.
* Check the main.cpp thoroughly to check what your system eventually has to be capable of. One of those functionalities will be reading a file. This file is included and may not be modified. You can open this file as “input.txt”.
* Write the needed classes. Make sure your classes have correct output() methods, the output method writes the classes to ‘std::cout’.
* Write the Order class which contains the order
* **Please note:** Make sure that by the end of the program all allocated memory gets deallocated when the program finishes! You should not leak any memory when the program terminates.
* **Please note:** Every class should have its own header file and cpp file. You should split out the functionality into different header and cpp files, so it is not allowed to put all source code in a single cpp file!

## Some hints for along the way

Because figuring out what to do can be tough, we have provided some additional hints so that you can see if you’re still going in the right direction:

* Test often. The application should be calling methods on some kind of Order class and you should print the results. Only implement the items that are asked within the TODO comments as this will ensure that your solution matches the output provided below.
* Stuck on a specific part? Try to work “around” the problem so you can still receive marks for the final product. Feel free to comment what part you were stuck on and how you tried to solve it.
* Make use of the fact that you can access the websites cplusplus.com en cppreference.com!  
    
  **Please note: The search function on cplusplus.com does not work, because google.com is blocked and the search function uses google.com!**
* As a reminder, realise we grade your product on:
  + Functionality and readability of the code
  + Proper class design (overall), the implementation of the individual classes and the proper use of inheritance and access specifier
  + Splitting the functionality into different header and cpp files
  + The use of STL algorithms and data-structures
  + The correct usage of exceptions
  + Correct allocation and deallocation of memory
  + Correct usage of method/function arguments (by-value, by-reference, by-address) and declaration of method const when possible.