1. Abstract
Dynamic Programming is used to solve many computationally complex tasks by dividing them into subproblems that are easier. The goal of this project was to develop software to help understanding the concept of dynamic programming by animating the solving process. We observed a problem with exponential time complexity and created a Python program for solving it while visualizing the entire process.

2. Introduction
As data amounts are growing order of magnitude faster than the processing power, all kind of algorithms giving us faster solutions or approximations of solutions are getting more important. Dynamic programming is one way to speed up the problem solving. It provides an advantage if the subproblems are overlapping. Usually a problem has to be solved by solving all the subproblems and combining them together for the final result. If the subproblems are overlapping then they are calculated multiple times. Dynamic programming avoids this problem by calculating each subproblem only once.

We also found out that understanding the general idea of dynamic programming is hard at first and a good visualization would probably help with the learning process a lot.

3. The analyzed problem
To help understanding the dynamic programming concept we used the problem proposed in the practical labs. The problem is about maximizing or minimizing an arithmetic expression of numbers that have the operations missing between the numbers. Operations that are allowed are multiplication and addition. Also brackets can be placed around sub expressions. For example the list of numbers is “2 1 3 4” and the maximal solution would be “(2+1)*3*4” which equals 36.
To tackle this task we used bottom-up dynamic programming. First solving the task on list of length of 2 and then increasing the subproblem size one by one, while using the previously calculated results, until entire problem is considered.

4. The reasoning
Interestingly no one from our practical lab session managed to solve previously described task correctly and use dynamic programming for it. That is when we noticed a general problem and decided to implement some kind of tool to help understanding the concept.

We used python for writing our application as it is a high level programming language and takes less effort than other languages to get the algorithm working. First we completed the homework task by implementing dynamic programming. Then we added the graphical user interface. It took a lot of time to come up with a good visual representation model for explaining the solution process intuitively. We hope this program helps next year students to grasp the idea more easily.

5. Program overview
Our application needs python (version 2.6) to run. After starting the program the user is asked to fill in the data input dialogue (Figure 1). Here it is possible to select if the algorithm should maximize or minimize the arithmetic expression. User has to also fill in the data that is used to solve the task. Data has to be numbers that are separated by comma. After inserting the data the user has to press the submit button or “Enter” key on the keyboard to proceed.

If the input is valid then the program starts the problem solution visualization (Figure 2). At the beginning the screen is empty and only the input data is shown. Now the program waits for user input to continue with the animation. By pressing any key the user can go to the next step of the solution process.

Figure 1: Data input screen.

Figure 2: Visualization start.
At each step a new possible solution for a subproblem is found (Figure 3). To find the best solution to a problem, all the possible combinations of subproblems that cover the same domain have to be considered. The animation shows which of the subproblems are used for the calculation step. The dark blue and dark red color mark the solutions of previously solved subproblems. If the subproblem is of length one then the initial data value is used. In case of using the previously solved subproblems, the light blue or light red indicates the input data that was used for solving that problem. In top of the window the current mathematical calculations are shown that are used for solving a subproblem. In case of maximization, the two subproblems are added and multiplied together and the maximum of the two is added to the list of possible solutions. The possible solutions are shown on the first line. The maximum of the solution candidates will be the solution for the observed problem. In case of minimization, the minimum of the addition and multiplication of the subproblems is chosen. After the solution candidates have been calculated the minimum of them is chosen as the solution for the observed problem.

The final solution for a subproblem is shown in green color (Figure 4). The input values that were used to calculate the green solution are marked with a light green color.

Program stops when the final solution of the entire problem is calculated (Figure 5). The solution is shown at the top of the solution pyramid in green color.
6. Summary
Our goal was to create an application in order to introduce the concept of dynamic programming (DP) to new students. We created a python program to animate the solution process of one specific problem with an exponential time complexity. Once understanding the idea behind DP there are a lot of problems that can be solved in the same manner. Hopefully our program helps to understand the technique of dynamic programming easier.

7. References
2. Application source code. www.joel.ee/aa (25.05.2011)