Algorithmic project++

krpsim

Summary: This project may be an algorithmic project, an operational research project, an AI project as well as an industrial project... As you like.

Version: 4

Contents

Chapter I

Preamble

The *REAL* recipe of puff pastry

Ingredients for 1kg of pastry:

- 500 g of flour
- 250 g of water
- 10 g of salt
- 375 g of butter or margarine

The recipe:

- Why don't you try to make your own puff pastry? For this recipe, set apart the ingredients for the basic dough.
- Pour the flour in the whisking bowl. If you don't have a mixer, you can use your hands but you should then pour the flour on your work plan.
- In either case, add the salt...
- ...and temperate water.
- Put the bowl on the mixer...
- ...set up the "hook" tool...
- ...mix the ingredients at medium speed for several minutes. When the flour is amalgamated...
- ...pour the dough on the work plan...

- ...make a ball with it. You will get a rather compact dough. If you're using your hands, knead the ingredients with the tips of your fingers slowly incorporating the flour to get a uniform dough.
- Whichever the method you chose, you will get to this point.
- Slit the top of your dough in a cross. It's now ready for "lamination".
- Wrap it in a film...
- ...and leave it to rest in a fridge for at least 30 minutes.
- The lamination: spread flour on your work plan...
- ...and put the dough in front of you.
- With your fingers, flatten the 4 corners of the dough...
- Reduce the dough while respecting the shape of the cross.
- The center should be thicker.
- Place the mild butter in the center. Make a rectangle out of it. It should be the size of the dough and should cover the whole center.
- Fold one side of the dough over the butter.
- Fold the opposite side.
- Then the right side...
- ...make sure all the edges are jointed.
- Take away the flour surplus...
- ...and fold the left side.
- Once the butter is trapped inside, hit the dough with a rolling pin so the butter is equally spread inside the dough. The dough should always keep a square shape.
- Reduce the dough in its length...
- ...but make sure the butter doesn't break through the dough. Create a long rectangle that should be 1 cm thick.
- Give the crust a 90° rotation...
- ...counter clockwise. Use the pin to flatten the corners that might be a little thicker.
- Fold the dough in three, starting with the right fold...
- ...then the left one.
- Position the different layers of dough side to side.
- Reduce the dough in its length again.

- Give it another 90° rotation counter-clockwise and fold in three.
- Right side first...
- ...Then the left one. Position the different layers of dough side to side.
- Mark the dough pressing two fingers into it. This means the dough has two rounds. It will help you remember if you want to take a pause or if someone else takes over.
- This detail is valid in a professional environment. Home, you will seldom have to work on several pastries at once. Wrap the dough and leave it to rest in a fridge for at least 30 minutes. Thus end the first two rounds. It will take six to achieve your puff pastry.
- Take your dough and lay it again on a floured work plan.
- Reduce the dough in its length again. Give the crust a 9O° rotation counter-clockwise. Never change the rotation direction.
- Fold the dough in three, starting with the right fold...
- ...Take away the flour surplus...
- Then the left side. Position the different layers of dough side to side.
- Reduce the dough in its length again.
- Spin the crust 90° counter-clockwise once again...
- ...fold in three, right side first...
- ...and the left one. Position the different layers of dough side to side.
- This time, mark with 4 fingers. Now you know why.
- Wrap your dough in a film. Put it in the fridge for at least 30 minutes. Your first four rounds are over! Four down, two to go!
- Take your dough and lay it again on a floured work plan.
- Reduce the dough in its length again. Give the crust a 9O° rotation counter-clockwise. Never change the rotation direction.
- Fold the dough in three, starting with the right fold...
- ...Take away the flour surplus...
- Then the left side. Position the different layers of dough side to side.
- Reduce the dough in its length again.
- Spin the crust 90° counter-clockwise once again...
- ...fold in three, right side first...
- ...and the left one. Position the different layers of dough side to side.

- This time, mark with 6 fingers. You know what this means? You're done!
- 30 more minutes in the fridge and you can use your puff pastry! You can keep it 3 to 4 days in the fridge and up to several weeks in the freezer but don't forget to wrap it to prevent it from crusting.

Chapter II

Objective

You may have seen a nice little project management graph showing several processes linked one to the other, depending on their respective dependance and constraint. Often, it's just a little number and the dependancies are simple, and you will just have to run through it once. But as soon as you have several more processes available, more choices, and some must be repeated, or the group is supposed to run for ever, it gets harder to find an optimized solution that will offer you the best performance.
With this project, you will have to do project and group management, but you will mostly have to optimize the performance of a process chain (a graph, actually), maximizing a result or reducing the timeframe as much as possible.

Chapter III

Mandatory part

Form a group of 2-3 persons. You will have to design a program that will read a file describing processes, analyze the whole, and propose a worthy solution. You will use this opportunity to create a second little verifying program, and at least one process file of your own. You will need it for you but also for the evaluation you will be responsible of.

The file format containing the process:

- A `#` to start each commentary line.
- A description of the available stocks in the beginning, in a simple `<stock_name>:<quantity>` format.
- A process description `<name>:(<need>:<qty>[;<need>:<qty>[...]]):(<result>:<qty>[;<result>:<qty>[...]]):<nb_cycle>` A process can start as soon as the stocks allow it. This can happen several times in the same cycle.
- One line only to indicate the elements that would need optimizing containing the key word `time` : `optimize:(<stock_name>|time[;<stock_name>|time[...]])`

Watch the example files provided with the subject.

The program `krpsim`:

- It is called `krpsim`
- You can choose your language. However, the use of a language, library or any external tool that would do the job for you is prohibited.
- It takes two: `krpsim <file> <delay>` parameters. The first one is the configuration file with stocks and process. The second one is the waiting time the program will not have to exceed (there will be some tolerance, here).

- You will also be free to choose your display, but it will have to be clear and let users understand the main actions of the program.
- Display will also have to create an output the verifying program can use following this format: `<cycle>:<process_name>`
- If resources are supposed to be consumed and the process is supposed to stop, display will have to run until the end. If the system is self-sustained, and runs forever, you will choose a reasonable stop condition and your stocks will have to show the process has ran several times.
- You don't have to set an invariance between two executions. For the same configuration, the proposed solution can be completely different from the first one.

The program `krpsim_verif`:

- It's called `krpsim_verif`
- It takes two : `krpsim_verif <file> <result_to_test>` parameters. The first parameter is the configuration file. The second one is a text file containing the `krpsim` trace that must be verified.
- Display must show the progress is correct or indicate the cycle and the process that are problematic. In any case, the stocks and the last cycle must appear at the end of the program.

Then, you will create at least one configuration file of your own and not just a copy-paste-replace of one of the provided files. Ideally, you should create 2 files. One that ends anyway once resources have been consumed, and another that can run forever.

Chapter IV

Bonus part

⚠ `Bonus will be taken into account only if the mandatory part is PERFECT. PERFECT meaning it is completed, that its behavior cannot be faulted, even because of the slightest mistake, improper use, etc... Practically, it means that if the mandatory part is not validated, none of the bonus will be taken in consideration.`

For this project the only bonus will be in relation to a perfect optimization of your program.

Have a good optimization!

Chapter V

Example

Here is a configuration file example:

```
#
# very simple demo - krpsim
#
# stock     name:quantity
euro:10
#
# process  name:(need1:qty1;need2:qty2;[...]):(result1:qty1;result2:qty2;[...]):delay
#
equipment_purchase:(euro:8):(equipment:1):10
product_creation:(materiel:1):(product:1):30
Delivery:(product:1):(happy_client:1):20
#
# optimize time for no process possible (eating stock, produce all possible),
# or maximize some products over a long delay
# optimize:(time|stock1;time|stock2;...)
#
optimize:(time;happy_client)
#
```

Example of possible output:

```
%> ./krpsim simple 10
Nice file! 3 processes, 4 stocks, 1 to optimize
Evaluating .................. done.
Main walk
0:equipment_purchase
10:product_creation
40:delivery
no more process doable at time 61
Stock :
 happy_client=> 1
 product => 0
 equipment => 0
 euro => 2
%>
```

Chapter VI

Submission and peer-evaluation

Turn in your assignment in your `Git` repository as usual. Only the work inside your repository will be evaluated during the defense. Don't hesitate to double check the names of your folders and files to ensure they are correct.