COS132 Programming **Practical #3** corresponding to the **Syllabus of the week 13.3.-16.3.**2023

Scenario:
This week *Antonio's Pizzeria Italiana* is closed for refurbishment. Moreover, Antonio also wants to use this time for improving his invoicing system, such that his customers will henceforth receive more accurate bills which display in more detail what base-pizza and how many additional toppings a customer has chosen. For this purpose, Antonio wants to introduce an **array** with seven fields, such that:

This field shows **how many** *small round base pizza* was ordered	This field shows **how many** *big rectangular base pizza* was ordered	This field shows **how many** additional *olive* toppings were ordered	This field shows **how many** additional *onion* toppings were ordered	This field shows **how many** additional *cheese* toppings were ordered	This field shows **how many** additional *salami* toppings were ordered	This field shows **how many** additional *shrimps* toppings were ordered

With this new data-structure in his software Antonio also wants the new program to run through two loops:
- In the **first loop**, a test-dummy customer (played by Antonio himself) most choose ten toppings onto some base-pizza (for testing purposes),
- In the **second loop**, which is the billing loop, a more precise invoice shall be generated on the basis of the data that had been collected in the **array** during the run of the above-mentioned first loop.

REQUIREMENTS SPECIFICATION

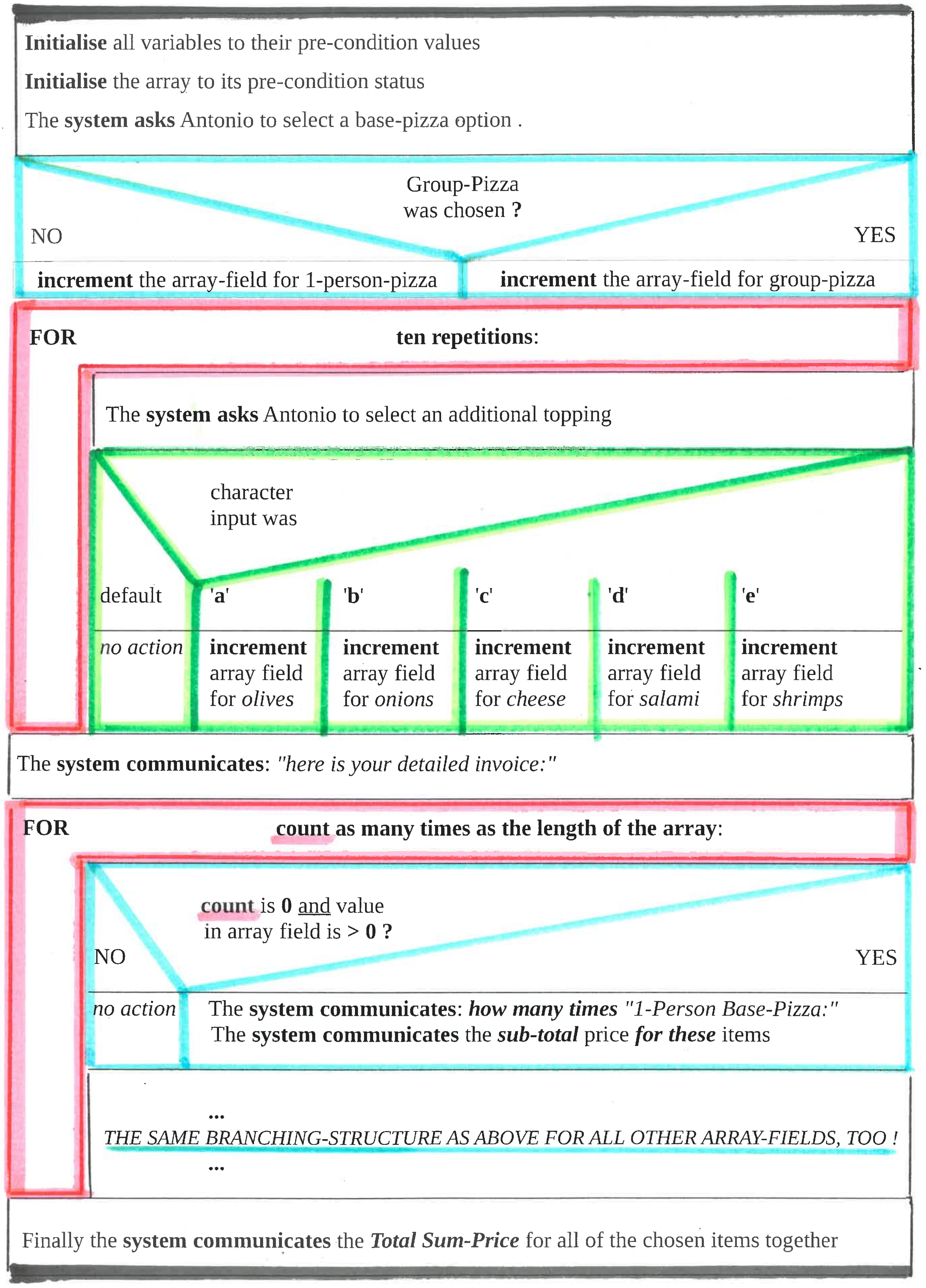
Pre-Conditions:
- constant: large rectangular base pizza for 3 people without additional toppings: 60,- Rand.
- constant: small round base pizza for 1 person without additional toppings: 40,- Rand.
- constant: additional_olives: 15,50 Rand.
- constant: additional_onions: 11,- Rand.
- constant: additional_cheese: 12,30 Rand.
- constant: additional_salami: 22,- Rand.
- constant: additional_shrimps: 25,40 Rand.
- ~~variable: budget~~ *// **No** budget, because Antonio only wants to test the system this week*
- variable: invoice: 0,- Rand. *// For the purpose of system testing*
- **array:** initialised to 0 in all of its fields.

ALGORITHM:
- See the **Nassi-Shneiderman Diagram** which is given → in the APPENDIX to this Requirements Specification document

Post-Condition:
After the algorithm has reached its *termination,* the detailed invoice *correctly reflects* the *selected* Pizza ingredients (as in the "run" of the algorithm).

YOUR TO-DO-TASKS:
- *Do not use advanced programming techniques were not yet shown in the Lectures!*
- **Follow the Algorithm** of the **given** Nassi-Shneiderman **Diagram**!
- **Implement** the given Requirements Specification *correctly* with a C++ program.
- **Test** your C++ carefully with **https://www.onlinegdb.com/online_c++_compiler**.
- **Ask a Tutor for help** in case that you get stuck with the problem.
- **Convince yourself that everything is OK before you submit** your work.
- **Submit** your thoroughly tested C++ program to the ClickUp submission website.

Do not miss the submission deadline!
Belated submissions will be rejected.
NO deadline-extention will be granted.


```
Initialise all variables to their pre-condition values
Initialise the array to its pre-condition status
The system asks Antonio to select a base-pizza option .
Group-Pizza was chosen ?
    NO:  increment the array-field for 1-person-pizza
    YES: increment the array-field for group-pizza
FOR ten repetitions:
    The system asks Antonio to select an additional topping
    character input was
        default: no action
        'a': increment array field for olives
        'b': increment array field for onions
        'c': increment array field for cheese
        'd': increment array field for salami
        'e': increment array field for shrimps
The system communicates: "here is your detailed invoice:"
FOR count as many times as the length of the array:
    count is 0 and value in array field is > 0 ?
        NO:  no action
        YES: The system communicates: how many times "1-Person Base-Pizza:"
             The system communicates the sub-total price for these items
    ...
    THE SAME BRANCHING-STRUCTURE AS ABOVE FOR ALL OTHER ARRAY-FIELDS, TOO !
    ...
Finally the system communicates the Total Sum-Price for all of the chosen items together
```