Knowledge-Based Intelligent System Development Report

Developed by Maxwell Twardowski and Ethan Coco

For appropriate context, our program, *kbiSys: Knowledge-Based Intelligent System* is written in Python using version 3.11.
The *kbiSys* system's structure consists of three main files; gui.py, parse.py, and logic.py. The GUI file (gui.py) is used to run the program by running 'python3.11 gui.py' in an appropriate Linux, Windows, or Mac terminal, assuming the current terminal directory is 'src' when attempting to run the program. The GUI file consists of all our graphical user interface elements using the library 'tkinter'. The GUI file communicates directly with both our logic file (logic.py) and our parsing file (parse.py). During runtime, when the 'Generate' button is pressed, assuming all input is formatted correctly, our GUI file first calls the parsing file to parse input provided in the textbox elements, the GUI file then calls the logic file's associated functions in order of the reasoning task(s) selected by the user, the GUI file sets these returned values to local variables to pass to the parsing file to parse output. Once the parsing file parses the output, it returns string values of the data to be outputted directly to the user, which is then outputted through a new 'output' window. In summary, our GUI file acts as a main file consisting of GUI elements and calls the other files to compute logic, the parsing file acts as a mediator for the GUI and logic files as it parses input and output to be used by each file.

For penalty logic, the program first parses the text box input by storing each conjunct statement into a list, with each disjunct statement in that list also being its own list. This leads to a list of lists, where each list index is a disjunct statement and all statements must evaluate to true as they are conjuncted together. After this list is created, the logic detects if any elements in the list are of type list, indicating that they are disjunct statements. It evaluates those disjuncts first using the current feasible object, and if any are false the penalty is automatically added as each disjunct statement needs to evaluate to true. If no elements are of type list, then the element is simply an attribute that must be true. If that attribute is in the current object, no penalty is added for it, but if it is not the penalty is added and the next penalty statement is checked. This leaves us with a dictionary of all total penalty values for each feasible object.

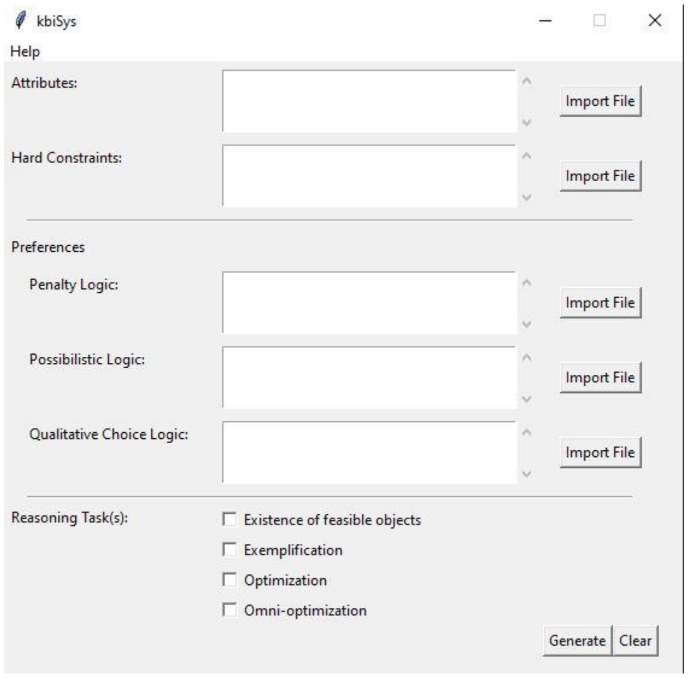
For possibilistic logic, it is largely the same; the program first parses the text box input by storing each conjunct statement into a list, with each disjunct statement in that list also being its own list. This leads to a list of lists, where each list index is a disjunct statement and all statements must evaluate to true as they are conjuncted together. After this list is created, the logic detects if any elements in the list are of type list, indicating that they are disjunct statements. It evaluates those disjuncts first using the current feasible object, and if any are false the possibilistic is updated. The possibilistic value begins at 1, indicating an ideal possibilistic, however the possibilistic is updated to take the minimum value of all possibilistic rules as the total possibilistic. If no elements are of type list, then the element is simply an attribute that must be true. If that attribute is in the current object, the possibilistic is not updated, but if it is not the possibilistic is updated and the next statement is checked. This leaves us with a dictionary of all total possibilistic values for each feasible object.

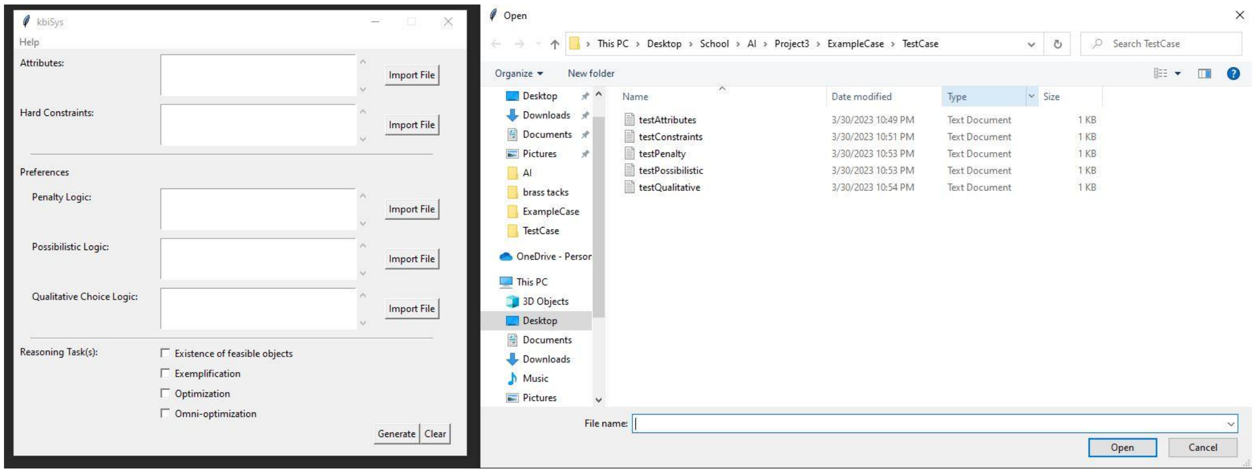
For qualitative choice logic, the function again first parses the text box input. The function then evaluates each rule, and stores the result for each rule in a list corresponding to the preference, with the most ideal rule being 1 and the least ideal being 999 to represent infinity. This will be done for each object, and we will be left with a list of lists, with each innermost list consisting of the preference values for each rule for each object. Each object is then checked against each other object to determine preference using Pareto ordering. This states that to be an optimal choice, the object must not be dominated by any other object in all objects being considered. This means that the object must have a weak preference over all other objects being considered. This is done by checking if an object is equal to another object in all preferences but better than an object in one, meaning the object is weakly preferred. The program tracks each object that gets dominated by another, and creates a list of dominated objects. The dominated objects are then removed from the list of feasible objects, resulting in a list of optimal objects that is then returned to the calling function.

After these logic functions have run and returned their respective outputs, the optimization function specified determines the optimal values. For penalty, it determines the smallest total penalty from the dictionary provided, and uses the resulting objects as the optimal. In the case of a tie, each object and their penalty are provided in a list. For possibilistic, the max is determined, and in the case of a tie each object and its possibilistic are provided in a list. For qualitative, there is already a list of optimal qualitative choices returned.

Screenshots of the program running at three different intervals are provided, for the test case that we have created:

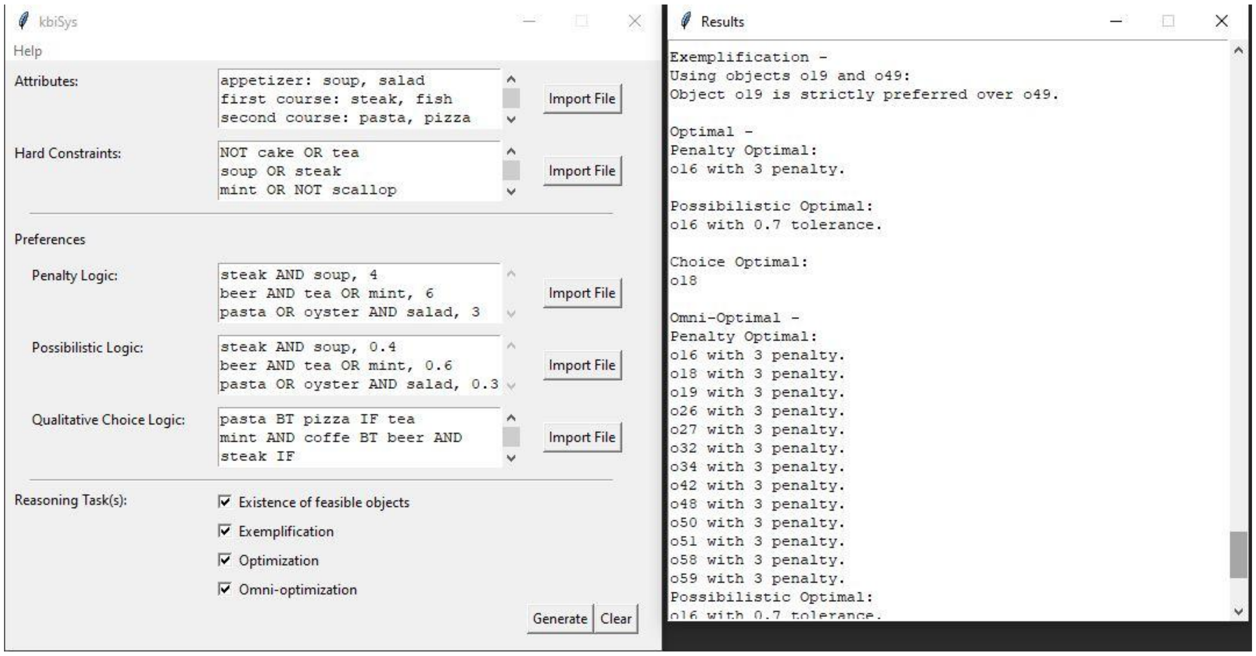
This is the GUI upon first starting the program, where all values are blank.





Name	Date modified	Type	Size
testAttributes	3/30/2023 10:49 PM	Text Document	1 KB
testConstraints	3/30/2023 10:51 PM	Text Document	1 KB
testPenalty	3/30/2023 10:53 PM	Text Document	1 KB
testPossibilistic	3/30/2023 10:53 PM	Text Document	1 KB
testQualitative	3/30/2023 10:54 PM	Text Document	1 KB

This is the GUI after clicking an import file button. The user is prompted to select a file for input, and can navigate their computer in a file explorer while doing so.



Attributes:
```
appetizer: soup, salad
first course: steak, fish
second course: pasta, pizza
```

Hard Constraints:
```
NOT cake OR tea
soup OR steak
mint OR NOT scallop
```

Preferences

Penalty Logic:
```
steak AND soup, 4
beer AND tea OR mint, 6
pasta OR oyster AND salad, 3
```

Possibilistic Logic:
```
steak AND soup, 0.4
beer AND tea OR mint, 0.6
pasta OR oyster AND salad, 0.3
```

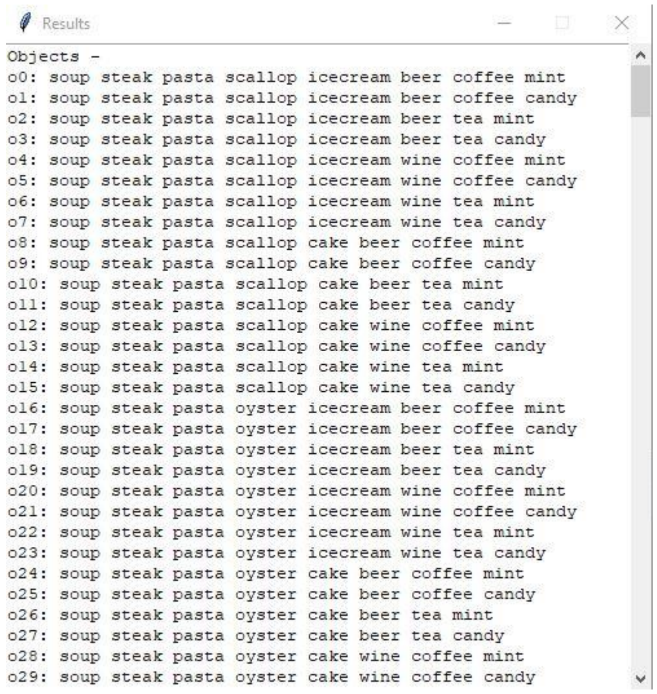
Qualitative Choice Logic:
```
pasta BT pizza IF tea
mint AND coffe BT beer AND
steak IF
```

Reasoning Task(s):
- [✓] Existence of feasible objects
- [✓] Exemplification
- [✓] Optimization
- [✓] Omni-optimization

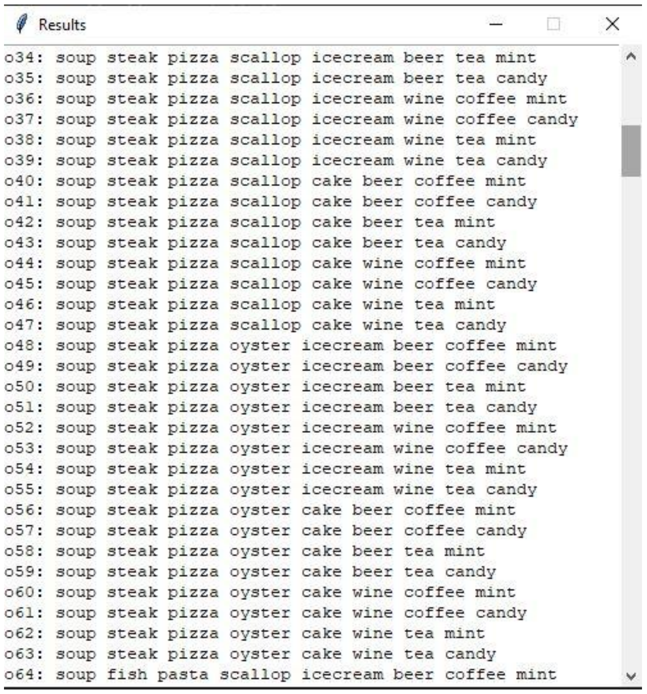
```
Exemplification -
Using objects o19 and o49:
Object o19 is strictly preferred over o49.

Optimal -
Penalty Optimal:
o16 with 3 penalty.

Possibilistic Optimal:
o16 with 0.7 tolerance.

Choice Optimal:
o18

Omni-Optimal -
Penalty Optimal:
o16 with 3 penalty.
o18 with 3 penalty.
o19 with 3 penalty.
o26 with 3 penalty.
o27 with 3 penalty.
o32 with 3 penalty.
o34 with 3 penalty.
o42 with 3 penalty.
o48 with 3 penalty.
o50 with 3 penalty.
o51 with 3 penalty.
o58 with 3 penalty.
o59 with 3 penalty.
Possibilistic Optimal:
o16 with 0.7 tolerance.
```

This is the GUI and output after running the program with our provided testing case. When a file is selected, the text is inserted into the text box. After all five files have been uploaded or the user has manually entered text into each box, the user has the option to select any number of the four reasoning tasks provided. If none are selected, the program will output all objects in line with our binary encoding algorithm. The example screenshot provided shows all four tasks selected, and our output contains a scroll bar that the user can use to navigate through the output if the output is lengthy. For this example case, objects are printed, then feasible objects are printed (they are higher up in the screenshot provided, note the location of the scroll bar on the right hand side), then the exemplification of two random feasible objects is shown, followed by a single optimal object for penalty, possibilistic, and choice, and then followed by all optimal values for penalty, possibilistic, and choice. Only a portion of output is displayed at a time for pleasant visualization, and the outputs that are not currently visible in this screenshot are all visible upon moving the scroll bar. If a reasoning task is not selected, that output will not display.

The following is all output for our example case provided, beginning with all objects, feasible objects, exemplification, a single optimal object, and ending with all optimal objects.

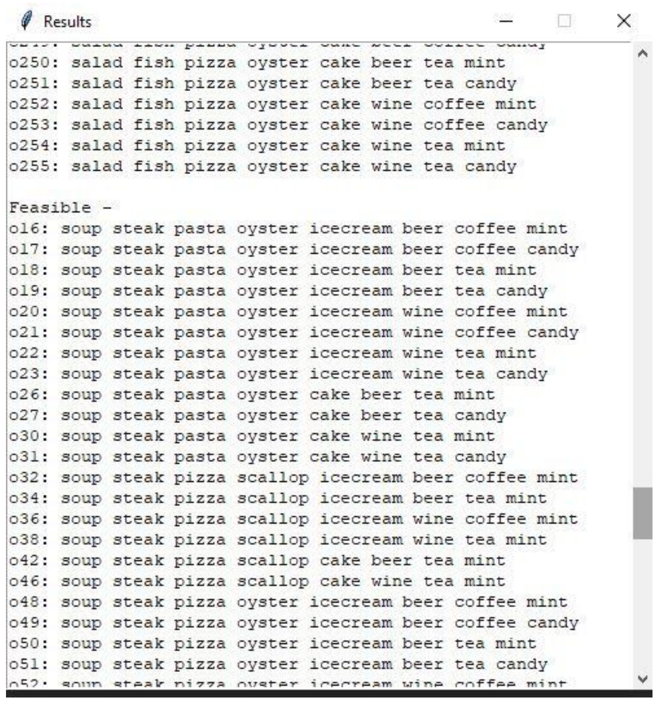


```
Objects -
o0: soup steak pasta scallop icecream beer coffee mint
o1: soup steak pasta scallop icecream beer coffee candy
o2: soup steak pasta scallop icecream beer tea mint
o3: soup steak pasta scallop icecream beer tea candy
o4: soup steak pasta scallop icecream wine coffee mint
o5: soup steak pasta scallop icecream wine coffee candy
o6: soup steak pasta scallop icecream wine tea mint
o7: soup steak pasta scallop icecream wine tea candy
o8: soup steak pasta scallop cake beer coffee mint
o9: soup steak pasta scallop cake beer coffee candy
o10: soup steak pasta scallop cake beer tea mint
o11: soup steak pasta scallop cake beer tea candy
o12: soup steak pasta scallop cake wine coffee mint
o13: soup steak pasta scallop cake wine coffee candy
o14: soup steak pasta scallop cake wine tea mint
o15: soup steak pasta scallop cake wine tea candy
o16: soup steak pasta oyster icecream beer coffee mint
o17: soup steak pasta oyster icecream beer coffee candy
o18: soup steak pasta oyster icecream beer tea mint
o19: soup steak pasta oyster icecream beer tea candy
o20: soup steak pasta oyster icecream wine coffee mint
o21: soup steak pasta oyster icecream wine coffee candy
o22: soup steak pasta oyster icecream wine tea mint
o23: soup steak pasta oyster icecream wine tea candy
o24: soup steak pasta oyster cake beer coffee mint
o25: soup steak pasta oyster cake beer coffee candy
o26: soup steak pasta oyster cake beer tea mint
o27: soup steak pasta oyster cake beer tea candy
o28: soup steak pasta oyster cake wine coffee mint
o29: soup steak pasta oyster cake wine coffee candy
```

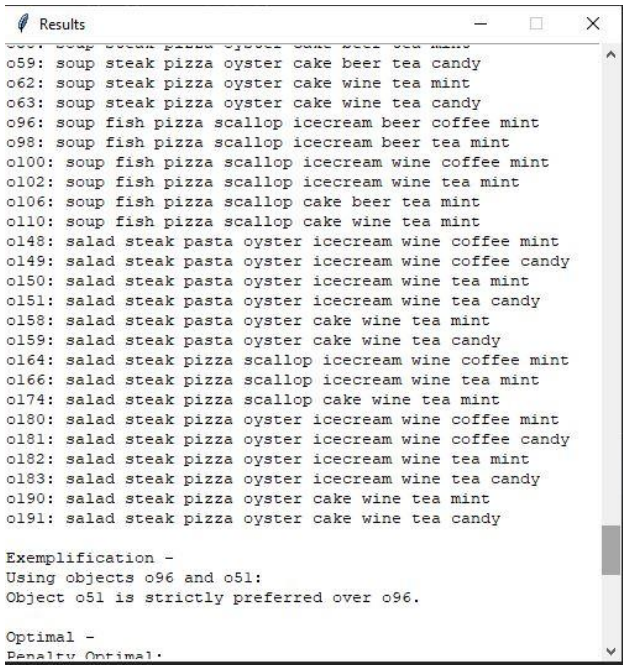


```
o34: soup steak pizza scallop icecream beer tea mint
o35: soup steak pizza scallop icecream beer tea candy
o36: soup steak pizza scallop icecream wine coffee mint
o37: soup steak pizza scallop icecream wine coffee candy
o38: soup steak pizza scallop icecream wine tea mint
o39: soup steak pizza scallop icecream wine tea candy
o40: soup steak pizza scallop cake beer coffee mint
o41: soup steak pizza scallop cake beer coffee candy
o42: soup steak pizza scallop cake beer tea mint
o43: soup steak pizza scallop cake beer tea candy
o44: soup steak pizza scallop cake wine coffee mint
o45: soup steak pizza scallop cake wine coffee candy
o46: soup steak pizza scallop cake wine tea mint
o47: soup steak pizza scallop cake wine tea candy
o48: soup steak pizza oyster icecream beer coffee mint
o49: soup steak pizza oyster icecream beer coffee candy
o50: soup steak pizza oyster icecream beer tea mint
o51: soup steak pizza oyster icecream beer tea candy
o52: soup steak pizza oyster icecream wine coffee mint
o53: soup steak pizza oyster icecream wine coffee candy
o54: soup steak pizza oyster icecream wine tea mint
o55: soup steak pizza oyster icecream wine tea candy
o56: soup steak pizza oyster cake beer coffee mint
o57: soup steak pizza oyster cake beer coffee candy
o58: soup steak pizza oyster cake beer tea mint
o59: soup steak pizza oyster cake beer tea candy
o60: soup steak pizza oyster cake wine coffee mint
o61: soup steak pizza oyster cake wine coffee candy
o62: soup steak pizza oyster cake wine tea mint
o63: soup steak pizza oyster cake wine tea candy
o64: soup fish pasta scallop icecream beer coffee mint
```

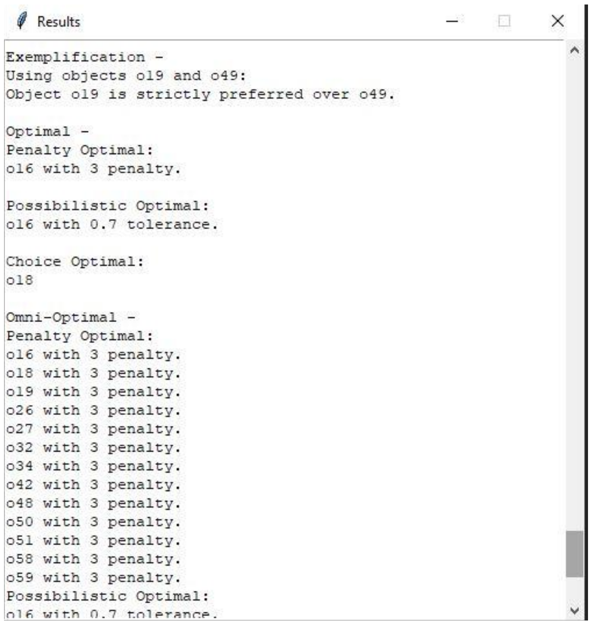
…



```
o250: salad fish pizza oyster cake beer tea mint
o251: salad fish pizza oyster cake beer tea candy
o252: salad fish pizza oyster cake wine coffee mint
o253: salad fish pizza oyster cake wine coffee candy
o254: salad fish pizza oyster cake wine tea mint
o255: salad fish pizza oyster cake wine tea candy

Feasible -
o16: soup steak pasta oyster icecream beer coffee mint
o17: soup steak pasta oyster icecream beer coffee candy
o18: soup steak pasta oyster icecream beer tea mint
o19: soup steak pasta oyster icecream beer tea candy
o20: soup steak pasta oyster icecream wine coffee mint
o21: soup steak pasta oyster icecream wine coffee candy
o22: soup steak pasta oyster icecream wine tea mint
o23: soup steak pasta oyster icecream wine tea candy
o26: soup steak pasta oyster cake beer tea mint
o27: soup steak pasta oyster cake beer tea candy
o30: soup steak pasta oyster cake wine tea mint
o31: soup steak pasta oyster cake wine tea candy
o32: soup steak pizza scallop icecream beer coffee mint
o34: soup steak pizza scallop icecream beer tea mint
o36: soup steak pizza scallop icecream wine coffee mint
o38: soup steak pizza scallop icecream wine tea mint
o42: soup steak pizza scallop cake beer tea mint
o46: soup steak pizza scallop cake wine tea mint
o48: soup steak pizza oyster icecream beer coffee mint
o49: soup steak pizza oyster icecream beer coffee candy
o50: soup steak pizza oyster icecream beer tea mint
o51: soup steak pizza oyster icecream beer tea candy
o52: soup steak pizza oyster icecream wine coffee mint
```

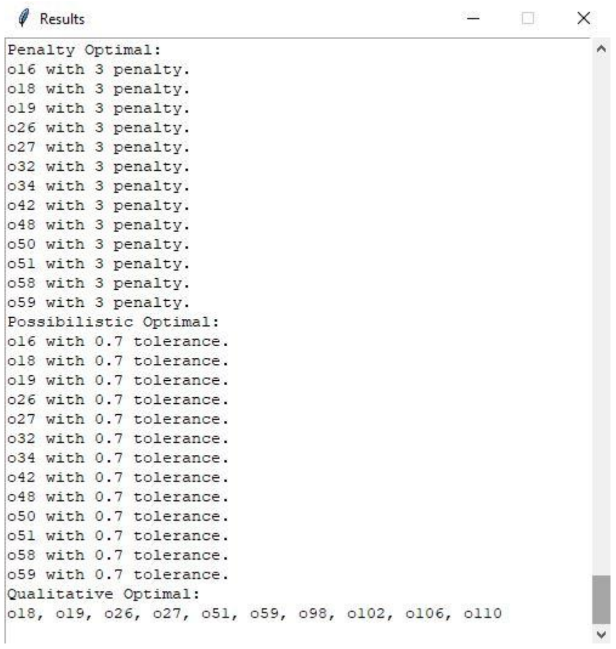


```
o59: soup steak pizza oyster cake beer tea candy
o62: soup steak pizza oyster cake wine tea mint
o63: soup steak pizza oyster cake wine tea candy
o96: soup fish pizza scallop icecream beer coffee mint
o98: soup fish pizza scallop icecream beer tea mint
o100: soup fish pizza scallop icecream wine coffee mint
o102: soup fish pizza scallop icecream wine tea mint
o106: soup fish pizza scallop cake beer tea mint
o110: soup fish pizza scallop cake wine tea mint
o148: salad steak pasta oyster icecream wine coffee mint
o149: salad steak pasta oyster icecream wine coffee candy
o150: salad steak pasta oyster icecream wine tea mint
o151: salad steak pasta oyster icecream wine tea candy
o158: salad steak pasta oyster cake wine tea mint
o159: salad steak pasta oyster cake wine tea candy
o164: salad steak pizza scallop icecream wine coffee mint
o166: salad steak pizza scallop icecream wine tea mint
o174: salad steak pizza scallop cake wine tea mint
o180: salad steak pizza oyster icecream wine coffee mint
o181: salad steak pizza oyster icecream wine coffee candy
o182: salad steak pizza oyster icecream wine tea mint
o183: salad steak pizza oyster icecream wine tea candy
o190: salad steak pizza oyster cake wine tea mint
o191: salad steak pizza oyster cake wine tea candy

Exemplification -
Using objects o96 and o51:
Object o51 is strictly preferred over o96.

Optimal -
Penalty Optimal:
```



Results

```
Exemplification -
Using objects o19 and o49:
Object o19 is strictly preferred over o49.

Optimal -
Penalty Optimal:
o16 with 3 penalty.

Possibilistic Optimal:
o16 with 0.7 tolerance.

Choice Optimal:
o18

Omni-Optimal -
Penalty Optimal:
o16 with 3 penalty.
o18 with 3 penalty.
o19 with 3 penalty.
o26 with 3 penalty.
o27 with 3 penalty.
o32 with 3 penalty.
o34 with 3 penalty.
o42 with 3 penalty.
o48 with 3 penalty.
o50 with 3 penalty.
o51 with 3 penalty.
o58 with 3 penalty.
o59 with 3 penalty.
Possibilistic Optimal:
o16 with 0.7 tolerance.
```



Results

```
Penalty Optimal:
o16 with 3 penalty.
o18 with 3 penalty.
o19 with 3 penalty.
o26 with 3 penalty.
o27 with 3 penalty.
o32 with 3 penalty.
o34 with 3 penalty.
o42 with 3 penalty.
o48 with 3 penalty.
o50 with 3 penalty.
o51 with 3 penalty.
o58 with 3 penalty.
o59 with 3 penalty.
Possibilistic Optimal:
o16 with 0.7 tolerance.
o18 with 0.7 tolerance.
o19 with 0.7 tolerance.
o26 with 0.7 tolerance.
o27 with 0.7 tolerance.
o32 with 0.7 tolerance.
o34 with 0.7 tolerance.
o42 with 0.7 tolerance.
o48 with 0.7 tolerance.
o50 with 0.7 tolerance.
o51 with 0.7 tolerance.
o58 with 0.7 tolerance.
o59 with 0.7 tolerance.
Qualitative Optimal:
o18, o19, o26, o27, o51, o59, o98, o102, o106, o110
```