Assignment week 12: ‘Pakjes-avond’ unbounded

Course ‘Imperative Programming’ (IPC031)

Make sure to start working on the assignment **before the lab** session, otherwise you will be too late for the deadline

1 Background

In this assignment you solve a search problem related to the most important Dutch event of the year: ‘pakjes-avond’ (gift evening).[1] Proceed as explained in the lecture, and first design the required data structures, then the recursive function prototype, followed by the preconditions and postconditions, and finally implement the base case and recursive case.

2 Learning objectives

After doing this assignment you are able to:

- implement search problems recursively;
- apply search problem solving techniques for new search problems;
- design appropriate data structures and function prototypes.

3 Assignment

Every year at ‘pakjes-avond’ St Nicholas faces the delicate problem of finding matching sets of gifts for the budget of each person. St Nicholas has a huge store of gifts. The name and price of each gift is recorded. You may safely assume that St Nicholas never runs out of gifts. However, you cannot assume that he has every possible gift. Miraculously, St Nicholas knows every person’s wish list and budget. A wish list contains names of gifts desired by a person, but without an order of preference.

Part 1: Design data structures

Design data structures to represent the gift store and wish lists. See Part 2 for a description of the data your data structures need to represent, and Part 3 to see how the data structures are used. You are allowed to use C++ constructs such as `structs` or `vectors` as (part of) your data structures.

Part 2: Getting data

On Brightspace, in “`assignment-12-mandatory-files.zip`” you may find a number of text files:

1. “`giftstore.txt`”: this file contains information about the gift store. Each line begins with the price of a gift (an integer value, representing the price in cents), followed by the name of the gift (which may contain white-space characters, *e.g.*, “Playstation 5”). This file contains all possible gifts by St Nicholas.
2. “`Andrew.txt`”, “`Belle.txt`”, “`Chris.txt`”, “`Desiree.txt`”, “`Edward.txt`”, “`Fabienne.txt`”: these files are wish lists. In a wish list file, the first line is a budget (an integer value, representing the budget in cents). All remaining lines are gift names (a single gift name per line).

Design and implement functions for reading these files into data structures introduced in Part 1.

Testing Verify that the loading of wish list files works correctly by implementing the “load” tests described in “`main.cpp`”.

[1] For more information about this event, read `https://en.wikipedia.org/wiki/Sinterklaas`.

Part 3: Compute the best gift set

Design and implement the recursive function:

```
int gifts (/* determine what function parameters you need yourself */)
```

For a given wish list and budget, this function computes a solution containing gifts that are on the wish list and are available in the gift store, and also returns the remaining amount of the budget. Each gift on the wish list is given at most once. In a solution the sum of values of the gifts is maximal with respect to the budget (obviously it may not exceed it). This means we may have multiple solutions, as multiple combinations of gifts could produce this same maximal sum.
The function must keep track of the best solution so far. When the function terminates, it should be possible to print the selection of chosen gifts and show how much money is left by printing the return value of the function.

Testing Design and implement the “compute_gifts” unit tests in “`main_test.cpp`” to test your functions.

Part 4: Putting it all together

Design and implement a `main` function that computes the best selection of gifts (use the `gifts` function from Part 3) for Andrew, Belle, Chris, Desiree, Edward, and Fabienne by reading the content of “`giftstore.txt`” and their wish lists (use the functions from Part 2). The remaining budget for them is:

- Andrew: 2 cents
- Belle: 102 cents
- Chris: 3 cents
- Desiree: 3 cents
- Edward: 3 cents
- Fabienne: 303 cents

Save the output of your program to a file called “`best_gifts.txt`”. Your program output must make clear which gifts each person receives.

4 Products

As product-to-deliver you upload to Brightspace:

- “`main.cpp`” that you have created with solutions for each part of the assignment.
- “`main_test.cpp`” that has been extended with the tests described in Parts 2 and 3, and non-trivial unit tests for each new function that you have developed in “`main.cpp`”.
- “`best_gifts.txt`” which contains the optimal gifts for each person.

Deadline

Lab assignment: Friday, December 6, 2024, 23:59h

Important notes:

1. check that you have actually submitted your solution in Brightspace.
2. the deadline is firm, and it is impossible to upload a solution after expiry. In that case you fail the assignment.
3. you can upload solutions to Brightspace several times, but only the last submission is stored.
4. identify yourself and your lab partner in every uploaded document. The identification consists of your first and last names, student numbers, and number of (sub) assignment. By identifying yourself, you declare that the uploaded work has been created solely by you and your lab partner.
5. your work will be judged and commented upon. We expect that you obtain the feedback, read it, and use it to for the next exercises.
6. it is essential that you only submit your own solution, never copy somebody/something else’s solution, and never share your solution—in particular: **AI tools (including but not limited to Github Copilot or ChatGPT) are not permitted**, solutions from previous year cannot be reused, and finally, you and your lab partner take joint responsibility for the assignment you submit.