Principles of Programming Language
Object Oriented Programming Division

Assignment

Draft Version 1.0

Instructions:

- All submissions must be made using Github for Classrooms. You can use your personal account or your IIIT-A account, as you wish. The class can be accessed by the URL:
  https://classroom.github.com/assignment-invitations/ecb4b642baa58522134642d00de209b1
  Support for the platform is provided by Vishnu KS (iit2013075@iiita.ac.in)

- The deadline for submission for q1 and q2 is 27th February, 2017, 10:00 AM. I will do a mass download for the same. You can continue to solve other questions as well and submit, and have an easy rest of semester.
- Please fill out the form to give the project URL. Please test that “git clone URL” works for the URL that you submit. https://docs.google.com/forms/d/1TrYIiloKIYVf2aalIvfsaS0SGd7GVuvKMwrUyGoF2mM/
- A plagiarism check will be performed.
- Please upload the class diagram as a PDF
- Use any tool for automatic documentation generation, which must also be supplied. Documentation is not only namespaces/class/function list, but also necessary comments.
- In a readme file describe the build system and how to compile/build the codes (preferably using command line only, without being forced to do an IDE import). Do not supply/upload pre-built libraries. Also mention the operating system, distribution and version on which the build was tested.
- Give commands to run the built examples (preferably using command line only).
- No GUI is required, though you may still create one. No Database is required.
- Use different namespaces for basic data items, data structures, algorithms and utilities.
- Even though collection frameworks/STL can be used, please create custom wrappers to have more suited operations as per questions rather than just the default library list.
- Each file should have one class, if using C++ the header file declaring the class should be different to the library file implementing the class.

Valentine’s Day is round the corner (at the time of framing this assignment) and hence the question. In a virtual world there exists some girls and some boys, with the number of boys far greater than the number of girls. A girl has a name, is rated by her attractiveness, has a maintenance budget associated with her and has an intelligence level. Each girl has a different criterion of choosing the boys to date, which can be either of: most attractive, most rich or most intelligent. However a girl will never date a boy whose girlfriend budget cannot meet her maintenance cost. The attractiveness for girls is already available by polls at anonymous websites, maintenance costs available from ex-boyfriends, and intelligence level from academic records.

Similarly every boy has a name, attractiveness, intelligence and budget. The budget is made publicly available, while the other metrics are known by similar mechanisms. A boy will never date a girl whose maintenance cost is more than his budget. Further, every boy has a minimum attraction requirements from a girl. Name can be taken as an ID and will never be common between any two people.

Every girl and boy is single to start with. Whenever a girl gets a boyfriend (or vice versa), both get committed. Neither may all girls be committed, nor may all boys be committed. In every such couple, the boy is expected to send gifts to the girl. Every gift has a *price* and a *value*. Both are known in advance as adjudged by experts. There are three types of gifts:

- *Essential Gifts*: These are bare minimal gifts and are associated with a price and value.
- *Luxury Gifts*: These gifts have the attributes of the luxury rating, difficulty to obtain the gift, value and price.
- *Utility Gifts*: These gifts are associated with the utility value, utility class, value and price.

As legend has it, there are three types of committed girls:

- *The choosy*, whose happiness in a relationship is logarithmic of the total cost of gifts achieved over maintenance. However the luxury gifts are very previous and count double the normal value.
- *The normal*, whose happiness in a relationship is linear to the total cost of gifts achieved over maintenance, including luxury gifts. The value of all gifts is added additional to the cost.
- *The desperate*, whose happiness in a relationship is exponential to the total cost of gifts received over maintenance, including luxury gifts. The value is not considered.

Similarly there are three types of committed boys:

- *The Miser*, who gift their girlfriend with enough gifts, equal or just over the maintenance cost. The happiness of these boys is given by the total unspent money from their budget.
- *The Generous*, who gift their girlfriend with maximum cost gifts, equal to or just under their budget. The happiness of these boys is given by the happiness of the girlfriend.
- *The Geeks*, who gift their girlfriend with enough gifts, equal or just over the maintenance cost. They additionally give one luxury gift, if budget allows. The happiness of these boys is given by the intelligence of their girlfriend alone.

While setting the gift basket is a DP/brute-force algorithm (subjected to integral or decimal costs), here a simple greedy algorithm will be used. From the list of gifts, starting from the cheapest, gifts will be added one by one, till the gifting constraints are satisfied. No gift can be repeated in the basket. If no gifting is possible, the boy can increase his budget so as to make a feasible gift basket.

The happiness of a couple is defined as the sum of the happiness of both girl and boy, accounting for all gifting taken place in the past. The compatibility of a couple is defined as the sum of: magnitude by which the budget of the boy exceeds the maintenance cost of the girl, the absolute value of the difference in attractiveness, and the absolute value of the difference of intelligence.

You need to make a logging utility to log (in a text file) all commitments, breakups and exchange of gifts. The logging should include timestamp, event type and the event description so as to be easily scanned while debugging.

You also need to make a testing utility in addition that creates random girls, boys and gifts of every type and stores it in CSV files (or any format that you like). These will be used by you for testing.

Solve the following questions. Each question should be an executable code file with minimal code. Any algorithm required for the same should be in the code library you make using the description above.

1. Allocate boyfriends to all girls in the same order as given in the input, and print the couples so formed without using inheritance.
2. After allocating boyfriends to all girls in the same order as given in the input, and after performing gifting to all girls as per logic, print the best *k* happiest couples and the best *k* most compatible couples. Also print all gift details of gift exchanges between all couples. Solve the question without using inheritance.
3. Solve q2 using inheritance (Suggestion: Make one collection for girls, one for boys and one for gifts, instead of 3 each)
4. Perform a breakup after Valentines day of the least *k* happy couples. Assign new (different) boyfriends to the girls who broke up.
5. Rather than making couples by taking girls one by one as in question 1, we create a different mechanism. First one girl chooses a boy, then one boy choses a girl, and so on. The girl list is ordered by maintenance cost, while the boy list is ordered by attractiveness. The boys always chose the most attractive available girl. Print the happiest *k* couples. Given programmer the choice between allotment algorithm between q1 and q4.

6. Suppose that gifting is not limited to Valentines day alone, but *t* days in a calendar month, well-marked. After every such day, couples with happiness less than *t* break up and form new couples.
7. You are given a list of boys. Find their girlfriend (if any). There are three implementations possible, storing committed boys in a couple as an array/linked list, sorted array (binary search) and hash table. All have their own pros and cons. Make all three implementations. The library should be such that the programmer of the main program should be able to choose any of the implementation or prefer not to choose in which case the default choice will be taken. (Suggestion: Implement all variants via inheritance and use upcasting to the base class).
8. Make a new gift allocation system, such that at least one gift of every type is given in the gift pack, even if it exceeds the budget. The logic should be implemented in the necessary library file and not in the main application program. Both types of gifting strategies should be implemented in the library, with a mechanism to specify the choice at the time of initialization. The default gifting type should be as stated previously, when no option is specified. (Suggestion: Make two GiftSelector classes inherited from a base class and upcasting to make the code generic)
9. Make a data structure that returns the best *k* valued items instead of the best valued item. Using this data structure make couples in q3 such that the boy picks up the girl by the secondary criterion (maintenance) instead of the primary criterion (attractiveness) from the top *k* alternatives. Similarly the girl picks up the secondary criterion instead of the primary (as per type) from the top *k* alternatives. Similarly the boy picks up a gift as per secondary criterion (value), instead of the primary criterion (cost) from the top *k* alternatives. Use templates to avoid creating different implementations for girls, boys and gifts. (Algorithm of any complexity is OK).
10. Make a data structure called as $Random_k$. The data structure instead of returning the highest value item, returns a random item out of best *n* items. This data structure should be used in q3 to make couples and select gifts. Use templates to avoid creating different implementations for girls, boys and gifts. (Algorithm of any complexity is OK).