Assignment – Christmas Basket of Cookies (100 pts)

Due December 1, 2019

Objectives: *This assignment gives you some experience with designing and writing C++ program using OOP features and STL classes: vector and list.*

- (5 points) Create a text file, called README, in which you provide:
  - Your First Name, Last Name, UIN, Section Number, User Name, E-mail address
  - State the Aggie Honor statement:

    I certify that I have listed all the sources that I used to develop the solutions and code to the submitted work.

    > *On my honor as an Aggie, I have neither given nor received any unauthorized help on this academic work.*

    Your Name Date
  - List any resources used such as webpages (provide URL). Do not mention the textbook and discussions with the Instructor, TA, or Peer Teachers.
  - List any known problems with the assignment you are turning in. For example, if you know your code does not run correctly, state that. This should be a short explanation.
  - Provide list of problems.
  - Provide a short description or pseudocode.
  - Write how you tested your program(s) for correctness and how you used exceptions for the error handling.
  - Submit to eCampus an electronic version of the file README along with your C++ code for the problems listed below and a hard copy to your TA.
  - The assignment will be graded focusing on: program design, correctness.
  - Your programs will be tested on a CSE machine.
  - Provide all tests done to verify correctness of your program in the report. Give an explanation why have you selected such tests.
  - Your program will be tested on a TA's input file.

Problem Description

1. Write a C++ program for organizing data and operations on this data. It contains items in no particular order and it might have duplicate items. It is probably the simplest way of organizing data, like for example a regular basket. You can think that your basket is filled with cookies of different colors and sizes and that there is no specific arrangement or organization of these items.

2. (25 points) Write a class `Cookie` which represents a Christmas cookie.

   (a) the class default constructor creates a cookie. Use only the following type of cookies: "`macadamia nut chocolate`", "`walnut chocolate chunk`", "`blackberry galettes`", "`white chocolate raspberry`" and sizes: `small`, `medium` and `large`. Use enum class `CookieType` for types and `CookieSize` for sizes.

   (b) the function `get_type()` returns the type of a cookie using the enum class `CookieType`

   (c) the function `get_size()` returns the size of a cookie using the enum class `CookieSize`

   (d) the operator `==(const cookie& c)` returns `true` if `c`'s type and size are the same as type and size of the cookie `this` (object calling the operator).

   (e) print a cookie as a pair of type and size in this form: (type, size). Example: `(macadamia nut chocolate, small)`

The class Cookie should be presented to your TA or PT during the labs by November 18, You should test all the implemented functions/operators of these class.

3. (10 points) Write an abstract C++ class `Basket` for the cookie basket which uses the class `Cookie`. The cookie basket holds cookies of different types and sizes (see above) and can contain many cookies of the same type and size. There are the following functions defined for a cookie basket:

   - add a cookie to the basket: `virtual void add_cookie(const Cookie& c)=0;`
   - check if a given type and size cookie is in the basket; return true if it is there and false otherwise: `virtual bool is_in_basket(const Cookie& c) const=0;`
   - remove and return a cookie (you have no control which cookie falls out): `virtual Cookie remove_any_cookie()=0;`
   - remove a cookie with a specific type and size from the basket: `virtual void remove_cookie(const Cookie& c)=0;`
   - make the basket empty: `virtual void clear_basket()=0;`
   - check if the basket is empty; return true if it is empty and false otherwise: `virtual bool is_basket_empty() const=0;`
   - return the total number of cookies in the basket: `virtual int cookie_total() const=0;`
   - return the number of cookies of the same size in the basket: `virtual int cookie_size_total(Cookie::CookieSize s) const=0;`
   - return the number of cookies of the same type in the basket: `virtual int cookie_type_total(Cookie::CookieType t) const=0;`
   - print the cookies in the basket (print cookie type and size, see the class `Cookie`) arranged with respect to their size: `virtual void print_cookies() const=0;`

4. (15 points) Design a derived class `Basket_Vector` which implements all the functions listed above. It defines a basket as a vector that can be automatically expanded dynamically as necessary using the C++ STL class `vector`.

The class Basket_Vector should be presented to your TA or PT during the labs by November 25, You should test all the implemented functions/operators of these class.

5. (15 points) Write a class `Basket_LinkedList` which implements all the functions listed above. It defines a basket as a linked list and you can use the C++ STL class `list` to support this implementation.

6. (15 points) Add these operations on baskets to the class `Basket` and implement them in the derived classes:

   - a copy operation that makes a copy of a basket: `virtual void copy_basket(const Basket& basket)=0;`
   - a merge operation that combines the contents of two baskets into a third basket (the contents of `basket1` and `basket2` are not changed): `virtual void merge_baskets(const Basket& basket1, const Basket& basket2)=0;`
   - an intersection operation that creates a third basket of those same cookies (of the same type and size) that are contained in two given baskets (the contents of `basket1` and `basket2` are not changed): `virtual void intersect_baskets(const Basket& basket1, const Basket& basket2)=0;`
   - a difference operation that creates a third basket of those cookies that are different (different type or size) in the first basket from those in the second one (the contents of `basket1 and basket2` are not changed): `virtual void difference_baskets(const basket& basket1, const basket& basket2)=0;`

   Specify each operation by stating its purpose, describing its arguments and by writing preconditions and post-conditions.

7. (15 points) Write a file `test_cookie_basket.cpp` where you will test the class `Basket` implemented based on `vector` and on `list`. For both the implementations (you can mix vector and list implementations):

   (a) Create two `Basket` objects. Fill them with cookies: read cookies from a user input file. Use between 15 and 25 cookies.

   (b) Print out their contents (`print_cookies()`). Print out the number of cookies of a given type and the total amount of cookies in the first and the second cookie basket.

   (c) Create a third `Basket` object which is the union of the two objects from (a). Print out its contents. Print out the number of cookies of the same type and size and the total number of cookies.

   (d) Print out the intersection of the objects from (a).

   (e) Print out the difference of the baskets from (a).