Programming

Assignment

This assignment involves a `Knapsack` class containing a function template to add objects of various sizes until the knapsack is near full.

General notes

1. Your assignment should be sensibly organised with the same kind of expectations in this area as the previous assignments, although you can directly include the cpp this time since it's templated.
2. If your code fails to compile you will likely receive zero.`$ g++ A3.cpp libGenerate.a -o KAP`
3. **Other than the initial command line input, the program should run without user input.**
4. You shouldn't modify `collect.h` and you do not need to submit it.

A knapsack

Knapsack problems relate to resource allocation. There is not going to any attempt to optimise in this case, you just add objects as they arrive, if they fit. There is a collection of classes A to G, provided in `collect.h` and you will need to pass instances of them to a knapsack in the order they arrive, until the next object cannot fit. You are to write a `Knapsack` class and the main to support and demonstrate the functionality required here. Your program should compile to KAP and run as:

./KAP size seed

- `size` : A positive integer. The size of the knapsack.
- `seed` : A positive integer. Random seed to be passed to the generate function.

A function `generate(int)` is prototyped in `collect.h` and defined in the library `libGenerate.a`. It returns a letter (char) that identifies which object you need to try and fit into your knapsack.

You need to pass an object of that type to the knapsack using a function template/template function defined inside `Knapsack`. That function template should take an object of arbitrary type and attempt to

"add it" to the knapsack. If the object fits, based on the size using `sizeof`, you record that object as being included, using the `name` attribute of the classes. The object itself should not be stored in the knapsack.

Once the next object to be passed cannot be added to the knapsack, you should stop generating objects and provide two reports:

- Knapsack size, fill size, and a list of object types in the order added:

```
Knapsack size: ...
Added object size: ...
BADACEGD
```

- A list of object types in alphabetical order with the size of each type and the number of each included:

```
A : size, 2
B : size, 1
C : size, 1
D : size, 2
E : size, 1
G : size, 1
```

Your reports shouldn't reference classes that haven't been added to the Knapsack and the Knapsack should never specifically reference the A to G types in `collect.h`, or their sizes. The `Knapsack` class should support the use of other types that contain a char accessible through a `getName()` member function.

Note that the `collect.h` and `libGenerate.a` can be changed for testing, so you shouldn't hardcode sizes or attempt to predict the output from `libGenerate.a`. Some example output, based on specific input, will be provided soon.

Please submit your source, so .cpp and .h files, and makefile if you have one, in a zip file `A3.zip`. There shouldn't be any directory structure within the zip file. Please don't submit `collect.h` or `libGenerate.a`.