School of Computing & Information Technology

CSCI251 Advanced Programming
Spring 2023

Assignment 3 (Worth 8%)

Due 11:55pm Friday in Week 13 (27th Oct,2023)

Overview

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

1. Organize your source files sensibly taking into consideration that you are using templates.
2. Compile your code into KAP using a command such as:
   $ g++ -std=c++17 -Wall A3.cpp libKnap.a -o KAP
3. Except the initial command line input, the program should run without user input.
4. You MUST not modify collect.h and you do not need to submit it. You are using it in developing and testing your code.
5. You are allowed to use vectors and maps to facilitate the implementation.
6. libKnap.a only works on Linux systems. So you mostly likely need to debug your code on Capa without sufficient help from IDEs, unless you are using Linux systems. This requirement is to let you get familiar with coding without being too dependent on IDEs.

A knapsack

Knapsack problems relate to resource allocation. In this assignment, the emphasis is not on optimization, you will 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 demonstrate the functionality required here. Your program should compile to KAP and run as:

./KAP size seed durability

- size : A positive integer. The size of the knapsack.
- seed : A positive integer. Random seed to be passed to the generate function.
- durability: A positive integer. The maximum number of resets the knapsack can execute.

A function generate(int) is prototyped in *collect.h* and defined in the library *libKnap.a*. It returns a letter (char) that identifies which object you need to try and fit into your knapsack. The generated letter can be A to G and R and S. The letters R and S are for special classes which trigger special condition 2 and 3. The special conditions will be described shortly.

You need to define a **function template** inside Knapsack and pass an object of that type to the knapsack using the function template. That function template should **take an object of arbitrary type as input (i.e., generic to objects of different types)** and attempt to "add it" to the knapsack. Note that the function template is allowed to take more than one input parameter if necessary. However, only one single parameter related to the object (i.e., the object itself) is allowed.

If the object fits, based on the size using the built-in function *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, which means that the actual object information is not stored. However, it takes some space of the knapsack which is equal to the returned value of sizeof(this object).** When one of the following conditions occurs, you need to output **a conditioned message** based on the condition we meet and then a **summary report**. The summary report and conditions with corresponding conditioned messages are described as below:

Summary report: It consists of two parts separated with "==========":

- Knapsack size, fill size, and a list of object types in the order added. For instance:
  ==========
  Knapsack size:
  Added object size: ...
  BADACEGD

- Information of objects in the knapsack based on the alphabetical order, with the size of each type and the number of each included. For instance:
  ==========

  A: size of an object A, 2
  B: size of an object B, 1
  C: size of an object C, 1
  D: size of an object D, 2
  E: size of an object E, 1
  G: size of an object G, 1
  ==========

Condition 1 (The next object (an object A to G) cannot be added to the knapsack due to size limit)

You should stop generating objects and output a conditioned message **followed by** a summary report with the format above.

The conditioned message is: "Condition 1 is triggered because the size of the next object is too large: $X". Here $X is the size of the next object.

Condition 2 (Attempt to add an R object)

The program produces a conditioned message **followed by** a summary report. After that, the Knapsack is reset, and all the previously stored objects are removed. If reaching the maximum number of resets, the process will be terminated. For instance, if the current reset is 4 and the durability is 10, **the conditioned message is as below**: "Condition 2: Attempt to add an R object

which triggers a reset. The number of used resets: 4 out of 10”

Condition 3 (Attempt to add an S object)

The program displays a conditioned message **followed by** a summary report, then stops the adding process. **The conditioned message is below**: “Condition 3: Attempt to add an S object which triggers early termination.”

Notice and Example outputs

Notice: The three conditions above are mutually exclusive; conditions 2 and 3 can be triggered without considering the size of R and S objects, and R and S objects cannot be added into the Knapsack; Objects not included in the knapsack should not appear in the summary report; The collect.h and libKnap.a can be changed for our marking, so you should not hardcode sizes or attempt to predict the output from libKnap.a.

Below are some examples. **Note that if you test your code on Capa, you should produce the same output given the same input.**

Example 1

```
shixunh@capa:~$ ./KAP 189 273 3
Condition 2: Attempt to add an R object which triggers a reset. The number of used resets:1/3
====================
Knapsack size: 189
Added object size: 144
ACAFAGADEE
====================
A : 1, 4
C : 4, 1
D : 8, 1
E : 16, 2
F : 32, 1
G : 64, 1
====================
Condition 2: Attempt to add an R object which triggers a reset. The number of used resets:2/3
====================
Knapsack size: 189
Added object size: 133
AGFFC
====================
A : 1, 1
C : 4, 1
F : 32, 2
G : 64, 1
====================
Condition 1 is triggered because the size of the next object is too large:32
====================
Knapsack size: 189
Added object size: 160
GDBAGECA
====================
A : 1, 2
B : 2, 1
C : 4, 1
D : 8, 1
E : 16, 1
G : 64, 2
====================
```

Example 2

```
shixunh@capa:~$ ./KAP 250 15 3
Condition 2: Attempt to add an R object which triggers a reset. The number of used resets:1/3
====================
Knapsack size: 250
Added object size: 220
GGBEAACGC
====================
A : 1, 2
B : 2, 1
C : 4, 2
E : 16, 1
G : 64, 3
====================
Condition 2: Attempt to add an R object which triggers a reset. The number of used resets:2/3
====================
Knapsack size: 250
Added object size: 190
EACGCFABBG
====================
A : 1, 2
B : 2, 2
C : 4, 2
E : 16, 1
F : 32, 1
G : 64, 2
====================
Condition 3: Attempt to add an S object which triggers early termination
====================
Knapsack size: 250
Added object size: 133
ACEFEG
====================
A : 1, 1
C : 4, 1
E : 16, 2
F : 32, 1
G : 64, 1
====================
```

Notes on submission

Submission is via Moodle.

Your code must compile on Capa with the instructions you provide.

Please submit your source (i.e. solution.cpp) file, Readme.txt file and makefile if you have one, in a zip file name_studentID_A3.zip**. There should not be any directory structure within the zip file.**

1. Late submissions will be marked with a 25% deduction for each day, including days over the weekend.

2. For source files, you only need to submit one single file named "solution.cpp" which contains the main function and Knapsack class. You MUST not modify collect.h and you do not need to submit it. You are using it in developing and testing your code.

3. Submissions more than four days late will not receive marks, unless an extension has been granted.

4. If you need an extension apply through SOLS, if possible **before** the assignment deadline.

5. Academic misconduct is treated seriously. Specifically, any plagiarised work will be awarded a zero mark and reported to the University. Be warned!