PROJECT TWO:

Costly Containers

Details

Summary:

- Worth: 10%
- Programming Due: 8:00am, Monday 20th October, 2008

Details:

- This project contributes 10% towards your final grade.
- Your files need to be submitted through the Assignment Drop Box anytime **before** 8:00am, Monday 20th October.
- Your **final mark** for this project will be based on the **style** and **correctness** of the code you submit

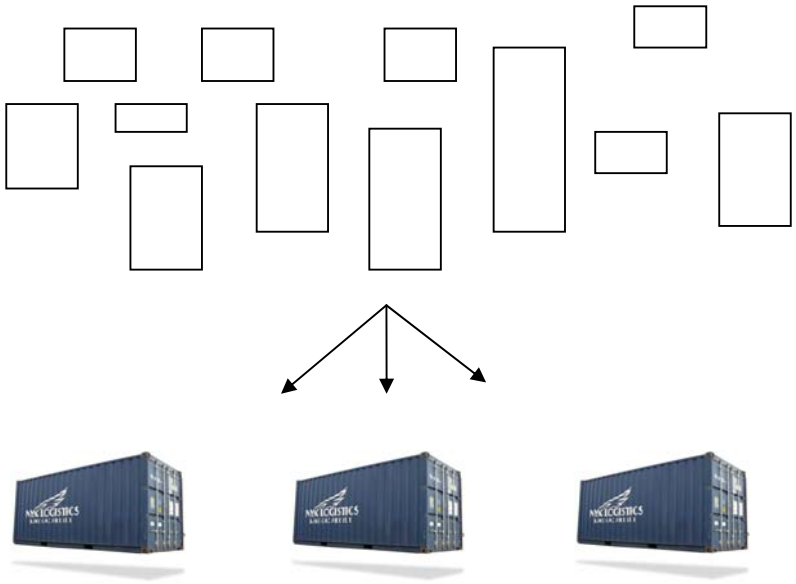
Resources:

- The "Projects" page of the course website contains skeleton code to help get you started
- The lecture on Wednesday 8th October is scheduled for specific discussion of this project

Description

For this project, you need to develop a program that determines whether it is possible to store a collection of items (of different sizes) in a given number of containers (each having the same capacity).

This application is to be used by an exporting company who regularly dispatch shipments of items, but pay high fees for the space taken up by the containers. Having your program, the company will be able to calculate the optimal allocation of items to containers and significantly reduce their annual costs.



Specifically, the input to your program will be the following:

- the number of containers
- the container capacity
- the number of items
- each of the item sizes

The output of your program must be the following:

- an allocation of items to containers, or a message indicating that this is not possible

Program input

The input to your program will be read from a file called `input.txt`. This file stores a number of sets of input data – your program must prompt the user for the particular data set that they would like to work with, and then read the data from that set from the file.

You can assume that the input data file, `input.txt`, will be located in the same directory as the source code for your program when compiled and executed from the command line.

The format of the file `input.txt` is simple, and is designed to allow multiple sets of data to exist in the same file. It can contain any text (such as comments or descriptions), but there is a special character that must only be used when defining a new data set.

This special character is: #

This character is only used to denote the beginning of a new data set. The first '#' character in the file will denote the beginning of the first set of data. The second '#' character in the file will denote the beginning of the second set of data, and so on.

The first data set is referred to as data set 1. The second data set is referred to as data set 2, and so on.

The '#' character is then followed by exactly three lines which contain a single integer, and a fourth line which contains a sequence of integers. An example of `input.txt` is given below:

input.txt

```
// Data file for EngGen 131 2008 Project Two

This is data set 1:
#
4
20
9
5 12 4 6 6 10 9 2 3

and this will be data set 2:
#
3
650
10
249 238 59 45 94 23 131 144 132 182
```

The values following the '#' character can be interpreted as follows:

- The first integer following the '#' character is the number of containers.
- The second integer following the '#' character is the container capacity.
- The third integer following the '#' character is the number of items.
- The fourth line contains each of the individual item sizes, separated with space characters.

The input file, `input.txt`, should be stored in the same directory as the source file. The following line of code should be used to open this input file in your program (where `fp` is a file pointer):

```
fp = fopen("input.txt", "r");
```

Program output

When your program runs, it must start by printing the following two lines of text:

```
Costly containers
EngGen 131 Sem 2 2008
```

and then it must prompt the user to enter the data set:

```
Which data set?
```

The user can then enter the data set they want to work with (shown in bold below):

```
Which data set? 2
```

Your program should then read the specified data set from file. In the example input file above, this means we would read the following data (from data set 2):

- the number of containers: 3
- the container capacity: 650
- the number of items: 10
- each of the item sizes: 249 238 59 45 94 23 131 144 132 182

This input data must be printed to the screen, as follows:

```
Number of containers: 3
Container capacity: 650
Number of items: 10
Item sizes: 249 238 59 45 94 23 131 144 132 182
```

The output from your program must be a valid allocation of the items to the containers. For example, the output given this data might be:

```
0: 249 238 59 45 23
1: 94 131 144 132
2: 182
```

Note that the number of lines of output in this allocation (3 lines in this example) is equal to the number of containers that were specified in the data set.

The example below shows what the output from your completed program should look like, if the user chooses data set 2 (using the example `input.txt` file given previously). User input is shown in bold:

```
Costly containers
EngGen 131 Sem 2 2008

Which data set? 2
Number of containers: 3
Container capacity: 650
Number of items: 10
Item sizes: 249 238 59 45 94 23 131 144 132 182

0: 249 238 59 45 23
1: 94 131 144 132
2: 182
```

If there is no solution (i.e. there is no way to allocate the set of items to the specified number of containers) then the output from the program should be:

```
There is no solution
```

Note, not all containers need to necessarily be used. In fact, for data set 1 in the previous `input.txt` file, it is possible to assign all items to just 3 of the containers. The output from the program, when using this data set, is given below:

```
Costly containers
EngGen 131 Sem 2 2008

Which data set? 1
Number of containers: 4
Container capacity: 20
Number of items: 9
Item sizes: 5 12 4 6 6 10 9 2 3

0: 5 12 2
1: 4 6 6 3
2: 10 9
3:
```

Container 3 is not required in the solution to this problem.

Checking your program before submission

The markers have a large number of projects to assess – unfortunately they do not have time to debug programs that you submit for marking. Do not submit code which does not compile.

When the markers are evaluating your program, they will be using a different "`input.txt`" file to the one that you have been working with. It is important that your program is correctly able to locate the input file on disk. If your program is not able to open the input file, then the marker will not be able to assess your program for correctness.

It is **strongly recommended** that you follow the steps below to ensure that your program will work correctly when it is being marked.

STEP 1:	Create an empty folder on disk
STEP 2:	Copy the source file for this project (`proj2.c`) into this empty folder. Copy the input file (`input.txt`) into this empty folder
STEP 3:	Open a command window (as described in Lab 7) and change the current directory to the folder that contains these two files
STEP 4:	Compile the source file using the command line tool, with the warning level on 4: `cl /W4 proj2.c` If there are warnings, you should address them before going on to step 5
STEP 5:	Run the program from the command line `proj2`

If you have followed the 5 steps above, and your program correctly reads data from the input file, then it should also work correctly for the marker.

Do not submit code which does not work having followed the 5 steps above – if the program cannot locate the input file in the same directory as the source file then the marker will not assess your program for correctness.

It is important that you use the following line of code to open the input file for reading:

```
fp = fopen("input.txt", "r");
```

Assessment and efficiency

A detailed assessment schedule will be distributed later, and discussed in the lecture on Wednesday 8th October. The most important thing is that your program generates correct solutions.

For certain input data sets, you may find that your program takes a long time to execute. There will be a small number of marks allocated for the speed at which your program executes. Data sets with which you can test the speed of your program will be distributed later.

Comments

At the top of your source file include a comment which clearly describes, in your own words, the purpose of your program.

You should also comment the rest of your code as necessary.

What to submit

You should submit the following source file for this project:

- `proj2.c`

This file should be submitted using the Assignment Drop Box:

```
https://adb.ec.auckland.ac.nz/adb/
```

before the deadline of 8:00am, Monday 20th October.