# 6215 Loadmaster

After a disaster, a rapid response is often necessary to airlift relief supplies into the affected area. The American Catastrophe Management Corporation has developed a set of cargo containers that fit into the seats of passenger aircraft. Each of these containers can be loaded onto and/or relocated within a plane in one minute. There are many fine programs that calculate cargo configurations when all the cargo weights are known prior to loading. Your team is to write a program that loads a specified aircraft as truckloads of containers arrive.

An airplane has two important loading limits: weight and balance. The takeoff weight (plane + fuel + cargo), or $TOW$, cannot exceed the airplane's specified maximum takeoff weight, or $MTOW$. In the interest of loading speed, your program will be given a threshold minimum takeoff weight, expressed as a percentage of MTOW, that should be achieved before considering the plane sufficiently loaded. Weights are specified as a whole number of pounds.

If the balance (center of gravity, or $CG$) of a plane falls outside a specified range along the longitudinal axis, the plane becomes aerodynamically unstable. The lateral CG of a traditional passenger plane is not a concern because the seats are close to the longitudinal axis; any side-to-side imbalance is corrected by trim adjustments. The minimum and maximum CG points are expressed as inches from the *datum*. The datum is an arbitrary point along the longitudinal axis. See Figure 1.

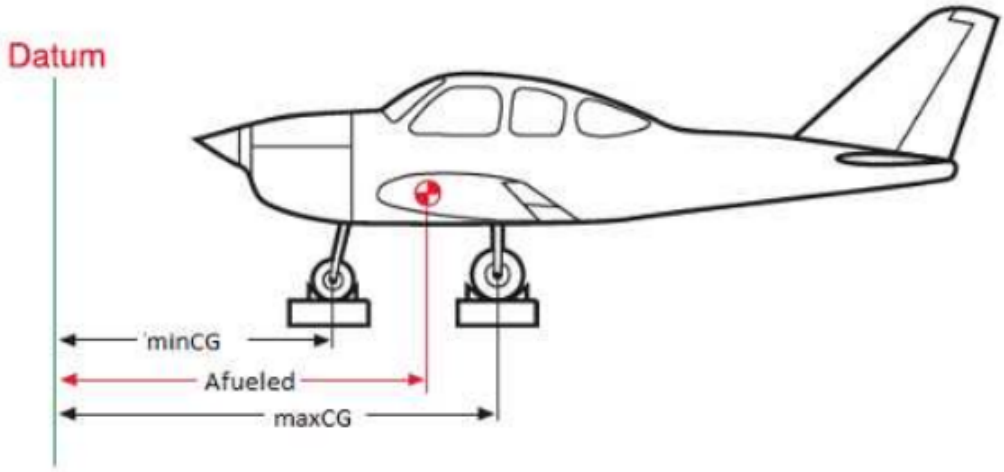


Figure 1. Diagram showing Datum and Center of Gravity parameters.

Often, the datum is a point in front of the aircraft so that CG computations don't involve negative numbers. The longitudinal CG of an airplane is determined from the Equation:

$$CG = \frac{TotalMoment}{TotalWeight}$$

where a Moment, $M$, is computed as $M = Arm \times Weight$. Arm is the distance from the datum, in inches, and Weight is in whole-number pounds. For example, Table 1 shows the center of gravity for a fueled plane with cargo weights A, B, C, and D at their respective arms. Note that the fueled plane is expressed as a weight with its arm at the CG of the unloaded plane.

| *Item* | *Weight (MTOW = 2200)* | *Arm* | *Moment* | *CG (+35 to +44)* |
|---|---|---|---|---|
| Airplane | 1340 | 37 | 49580 | |
| A | 352 | 35 | 12320 | |
| B | 212 | 72 | 15264 | |
| C | 240 | 48 | 11520 | |
| D | 50 | 92 | 4600 | |
| *Total* | 2192 | | 93284 | 42.5 |

**Table 1.** Sample loading and CG computation.

Your program must produce a safely loaded plane ($TOW \leq MTOW$ and $minCG \leq CG \leq maxCG$) when *one or more* of the Ready Conditions specified below are met:

a. $TOW \geq threshold/100.0 \times MTOW$

b. all seats are loaded with containers

c. no additional trucks arrive after $D_{max}$ minutes of having nothing to load (the need for relief exceeds the desire for efficient loading)

d. loading the lightest remaining unloaded cargo container would push $TOW > MTOW$.

**Program Interaction**

Your program must converse with a server, issuing commands to standard output and receiving responses from standard input. Your program starts the conversation by querying the airplane configuration. After receiving the configuration from the server, your program then repeatedly reads a timed event and responds to it. The first command is '*C*', a single character followed by end-of-line. The server replies with the airplane configuration:

$MTOW\ W_{fueled}\ A_{fueled}$
$threshold\ D_{max}$
$minCG\ maxCG\ P$
$a_1$
...
$a_P$

The first configuration line contains $MTOW$, $W_{fueled}$, and $A_{fueled}$, separated by whitespace. $MTOW$ is the maximum takeoff weight, $W_{fueled}$ is the weight of the fueled plane (plane + fuel), and $A_{fueled}$ is the moment arm of the fueled plane. The second line contains threshold and $D_{max}$, where threshold is the integer percentage of $MTOW$ for minimum loading. $D_{max}$ is the maximum idle delay to wait before deciding that no more cargo is arriving. Once your program decides not to load any further containers, $D_{max}$ is the maximum number of minutes allowed to reshuffle the contents within the plane to achieve balance. The third line contains $minCG$ and $maxCG$, the minimum and maximum center of gravity arms (in inches) and $P$, the number of passenger seats in the airplane. Following the third line are $P$ additional lines, each specifying the loading arm of each seat, expressed as whole-number inches from the datum.

Implicit in the order of the seat arms is the seat number.

After the configuration, your program reads an event from the server. An event is of the form

$t\ [C$
$w_1$
...
$w_C]$

where $t$ is the current step time in minutes. If a truck arrives, $C$ will be present, followed by $w_1$ through $w_C$ weights, expressed as whole-number pounds. Implicit in the ordering of the cargo weights is the cargo container number. Each truck starts again with its own items 1..$C$.

Your program responds to an event with one of the following commands.

| *Command String* | *Discussion* |
|---|---|
| L $c_i$ $p_j$ | Load cargo item $i$ into seat position $j$. |
| M $p_i$ $p_j$ L $c_m$ $p_n$ | Move cargo from seat $p_i$ to seat $p_j$ before loading cargo item $c_m$ into seat $p_n$. |
| M $p_i$ $p_j$ | Move cargo from seat $p_i$ to seat $p_j$. In the interest of loading speed, your program can perform a move without load only when the current truck was emptied and no additional truck has arrived yet. |
| I | Idle. Do nothing. |
| A $p_i$ $p_j$ | Cease loading items onto the plane, but make final adjustments. The Adjust command is almost the same as Move, but Adjust prevents the server from forcing Loads or MoveLoads while unloaded cargo items remain. Because no cargo items are being loaded, your program may move a cargo container to and from the aisle (position 0) to perform swaps. Once the A command is issued, your program may issue only the A or R commands. |
| R | Ready. At least one of the Ready Conditions specified above has been met. |

It is very important that your program flush standard output after issuing a command to the server.

e.g.

```
   C  fputs("C\n",stdout); fflush(stdout);
 C++  cout << "C\n" << flush;
Java  System.out.println("C"); System.out.flush();
```

Some important considerations:

- Only one plane can be loaded at a time (different planes require separate execution runs).
- Containers can only be loaded from one truck at a time.
- The number of passenger seats on an airplane varies from 4 to 853.
- The maximum number of cargo containers in a truckload is 100.
- Once a container is loaded, it cannot be unloaded to achieve safe weight or balance.
- Although some planes carry passengers on more than one level, the vertical relationship of seats to the datum is ignored, and thus not specified.
- It will always be possible to achieve one or more of the Ready Conditions.

**Sample Transcript**

A sample transcript follows. Client commands are offset for clarity. The comments to the right of the transcript must not appear in your client commands and will not occur in the server responses; they are for description only.

| *Server* | *Client* | *Comments* |
|---|---|---|
| | `C` | (Query configuration) |
| `2200 1340 37` | | (*MTOW* $W_{fueled}$ $A_{fueled}$) |
| `90 3` | | (*threshold percentage* and $D_{max}$) |
| `35.0 44.0 4` | | (*minCG maxCG* and $P = 4$ passenger seats) |
| `35` | | ($a_1 = 35.0$ inches) |
| `72` | | ($a_2$) |
| `48` | | ($a_3$) |
| `92` | | ($a_4$. This is the end of the configuration.) |
| `1 2` | | (The first event will always be a truck arrival at $t = 1$.) |
| `240` | | ($w_1 = 240$ pounds) |
| `50` | | ($w_2$) |
| | `L 1 1` | (program loads cargo item 1 (240 lbs.) to seat position 1 at 35 inches) |
| `2` | | (event at time = 2, no truck can arrive while unloaded containers remain) |
| | `L 2 3` | (cargo item 2 (50 lbs.) to seat position 3 at 48 inches) |
| `3` | | (event at time = 3, nothing to load, waiting for a truck) |
| | `I` | (do nothing while waiting) |
| `4` | | (event at time = 4, nothing to load, still waiting for a truck) |
| | `I` | |
| `5 2` | | (at time = 5, another truck arrives with two containers) |
| `352` | | ($w_1 = 352$ pounds) |
| `212` | | ($w_2$) |
| | `M 3 2 L 1 3` | (move cargo from seat 3 to 2, load cargo item 1 (352 lbs.) to seat 3) |
| `6` | | |
| | `M 2 4 L 2 2` | (move cargo from seat 2 to 4, load cargo item 2 (212 lbs.) to seat 2) |
| `7 1` | | (at time = 7, another truck arrives with one container) |
| `500` | | ($w_1 = 500$ pounds) |
| | `R` | (at least one Ready Condition satisfied) |

**Judged Responses**

An ill-formed command produces a PRESENTATION ERROR. If the server has to wait longer than one second (real time) for your client to issue a command, the judged response is TIME LIMIT EXCEEDED. Any other problem with your client will result in a response of WRONG ANSWER.

**Hints for Testing**

You may construct a series of simulated events manually. Execute your client program with you, the programmer, acting as the server. Let standard input come from your command line environment. Supply correct responses to you program, keeping track of cargo containers and where they are loaded.

The server process used by the judges is available to the contestants. You may use the server in two ways. In the absence of a working client program, you, the programmer, can act as the client, typing input to the server. Use the following command:

`test7` *configurationFile transcriptFile*

To test your program and its conversation with the server, use the command:

`test7` *configurationFile transcriptFile sourceFile*

where:

- *configurationFile* is described below
- *transcriptFile* is a file that captures the server's inputs and outputs for later analysis. The server's inputs are your client's commands. Client commands are offset by two tab characters to distinguish the client vs. server dialog.
- *sourceFile* is the source code to your client. When specified, the `test7` script will call the compile script before executing your program.

You can produce a simulation to the server by supplying a configuration file in the following format:

The first three lines contain whitespace separated fields: line one contains $MTOW$, $W_{fueled}$, and $A_{fueled}$; line two contains $threshold$ and $D_{max}$; and line three contains $minCG$, $maxCG$, and $P$.

The following $P$ lines contain moment arm locations, expressed as whole number inches, for each of the $P$ seats. The first $3 + P$ lines are exactly the information presented in response to a configuration query.

Following the airplane configuration information are truck arrival events of the form:

$t_a$ $C$
$w_1$
...
$w_C$

where $t_a$ is the arrival time in minutes of a truck at the head of the line for loading. The first arrival event must be at $t_a = 1$. $C$ is always $\geq 1$. $C$ lines follow with one cargo weight per line. Subsequent arrival events must obey the condition $t_{a+1} \geq t_a + C$. The configuration for the above transcript is as follows. The comments to the right of the configuration file entries must not appear in the actual file; they are here for information only.

```
2200 1340 37    (MTOW W_fueled A_fueled)
90 3            (thresholdPercentage and D_max)
35.0 44.0 4     (minCG maxCG and P = 4 passenger seats)
35              (a_1 = 35.0 inches)
72              (a_2)
48              (a_3)
92              (a_4)
1 2             (at time = 1, a truck arrives with two containers, w_1 = 240, w_2 = 50)
240
50
5 2             (at time = 5, another truck arrives with two containers, w_1 = 352, w_2 = 212)
352
212
7 1             (at time = 7, another truck arrives with one container, w_1 = 500)
500
```