Unreliable, V1

Contents

Dealing with allocation failures

In lab 4, your code will be subject to allocation failures. Instead of calling malloc or calloc, it will call modified versions found in yet more libraries. The reliable library works as expected. The unreliable library fails at "random" times, causing balls and nodes to fail to allocate. Your code must gracefully deal with these events as described here.

Recovery

Your code should not quit on memory failure. If a ball fails to allocate, return to the input loop and carry on. If a node fails to allocate, insert returns false and the lab 4 code will have to free the allocated ball since the list doesn't have a node to hold it. This is *a* recovery strategy and not *the only* recovery strategy. It presumes that our code took measures to make future allocations work and it presumes that rolling over and playing dead at the first sign of difficulty is not acceptable. The Apollo 11 Eagle spacecraft went into computer overload on descent to the moon but the software was built to keep the important stuff alive even if it meant the other stuff starved. One of the questions to ask when you are given a programming assignment is, "what is our error recovery strategy?"

Details

Code Changes

Change every malloc, calloc, and free call to use the "alternative" equivalents instead.

Note that the "libraries" are not actual libraries. They are .o files that have been compiled with –g so that you can run gdb without issues. Here is the altmem.h header file found in the altmem.zip archive in piazza:

```
/* alternative memory calls (both reliable and unreliable use these) */
void *alternative_calloc(size_t nmemb, size_t size);
void *alternative_malloc(size_t size);
void alternative_free(void *ptr );
```

You will want to

```
#include <stdlib.h>
#include "altmem.h"
```

in the 2 files you have that deal in dynamic memory. Any file that includes altmem.h must *first* include stdlib.h. Note also that other header files might require stdio.h. When including files, put standard headers <stdio.h> before custom headers "altmem.h" [note the <> versus ""].

There will be point deductions if any of the non-prototype code you turn in directly calls malloc, calloc, or free.

Your dynamic memory code should be restricted to one file on the application side (this is a lab 3 mandate). Likewise your linkedlist.c file should be the only file that deals in dynamic memory for nodes.

What follows is example makefile lines suitable for lab4. It builds the lab4 target – a mandatory target – using the reliable memory allocation code. The lab4**u** target builds against the unreliable memory allocation code. Note the highlighted items. Modify the dependency list to suit, but leave either reliable or unreliable in there (but not both).

```
#Here are some example lines from a lab 4 makefile prior to writing linked list code

lab4u: lab4.o bits.o input.o memory.o output.o physics.o sim.o n2.o unreliable.o
        gcc -g -o $@ $^  -lm -L. -lpb -lncurses -llinkedlist

lab4: lab4.o bits.o input.o memory.o output.o physics.o sim.o n2.o reliable.o
        gcc -g -o $@ $^  -lm -L. -lpb -lncurses -llinkedlist
```

Prototypes for altmem

The prototypes below are merely tests if you use the entire lab code base.

Suggested prototype / test: Take your existing code; add the word `alternative_` in front of all calloc, malloc, and free calls. Then adjust your makefile to add reliable.o to the rule that builds that code. Then make and test.

Suggested prototype / test: Create a nearly identical target to the prototype above in your makefile. Change reliable.o to **un**reliable.o and test. Use this to test code you wrote in lab 3 that never got tested because malloc and calloc never failed. Get those bugs out early. The unreliable allocator fails on the

first call, the third call, and the sixth call… So you should be able to test the 2 test cases: fails to allocate a ball and fails to allocate a node after a successful allocation of a ball.

Initial testing for allocation failures

In the table below it shows the sequence for a file with multiple balls (x7.pbd). The left column will have allocation failures for both balls and nodes. The right column only has ball allocation failures since the library list code will not fail to allocate nodes.

As balls leave the table, it is possible that no node will be available on the off-table list and the ball will need to be freed.

Unreliable, with student written list code:	***Unreliable, with library list code:***
The first ball fails to allocate	The first ball fails to allocate
The second ball allocates	The second ball allocates and so does its node
The in-play list node for that ball fails to allocate	The third ball fails to allocate
The third ball allocates	The fourth ball allocates and so does its node
The in-play list node for third ball allocates, putting one ball total on the table.	The fifth ball allocates and so does its node
The fourth ball fails to allocate	The sixth ball fails to allocate
The fifth ball allocates	The seventh ball allocates and so does its node
The in-play list node for the fifth ball allocates, putting a second ball onto the table.	
The sixth ball allocates	
The in-play list node for the sixth ball fails to allocate	
The seventh ball allocates	
The in-play list node for the seventh ball allocates, putting a third ball on the table. (At this point the sim rums)	
Ball 9A comes off the table and the node for it on the off table list allocates	
At the same time, ball AC comes off the table and the node for it on the off-table list allocates	
The last ball (BE) comes off the table, but the node in the off-table list fails to allocate	

Your code is **required** to generate error messages when allocation fails. Here are examples using x7.pbd. These show the initialization sequence in its entirety and the unreliable one shows a later error as well.

Unreliable, coded with the linked list library:

```
DIAGNOSTIC: Successfully opened x7.pbd for reading.
Loaded     Red 08 1 ball at   0.00000,   0.00000  45.00000 deg  75.00000 ips
ERROR: failed to allocate ball
Loaded   Green 11 2 ball at   0.00000,   0.00000  60.00000 deg  75.00000 ips
DIAGNOSTIC: 1 balls allocated
DIAGNOSTIC: 1 nodes allocated.
Launched   Green 91 2 ball at   0.00000,   0.00000 at  37.50000,  64.95191
Loaded  Yellow 1A 3 ball at   0.00000,   0.00000  75.00000 deg  75.00000 ips
ERROR: failed to allocate ball
Loaded    Blue 23 4 ball at   0.00000,   0.00000  90.00000 deg  75.00000 ips
DIAGNOSTIC: 2 balls allocated
DIAGNOSTIC: 2 nodes allocated.
Launched    Blue A3 4 ball at   0.00000,   0.00000 at   0.00000,  75.00000
Loaded Magenta 2C 5 ball at   0.00000,   0.00000 105.00000 deg  75.00000 ips
DIAGNOSTIC: 3 balls allocated
DIAGNOSTIC: 3 nodes allocated.
Launched Magenta AC 5 ball at   0.00000,   0.00000 at -19.41143,  72.44444
Loaded    Cyan 35 6 ball at   0.00000,   0.00000 120.00000 deg  75.00000 ips
ERROR: failed to allocate ball
Loaded   White 3E 7 ball at   0.00000,   0.00000 135.00000 deg  75.00000 ips
DIAGNOSTIC: 4 balls allocated
DIAGNOSTIC: 4 nodes allocated.
Launched   White BE 7 ball at   0.00000,   0.00000 at -53.03301,  53.03301
Final scanf call returned -1
```

Unreliable, coded with alternative memory in your linked list code:

```
DIAGNOSTIC: Successfully opened x7.pbd for reading.
Loaded     Red 08 1 ball at   0.00000,   0.00000  45.00000 deg  75.00000 ips
ERROR: failed to allocate ball
Loaded   Green 11 2 ball at   0.00000,   0.00000  60.00000 deg  75.00000 ips
DIAGNOSTIC: 1 balls allocated
ERROR: linkedlist.c: Failed to malloc a Node
ERROR: Unable to insert into in-play list, freeing 91
DIAGNOSTIC: 1 balls freed
Loaded  Yellow 1A 3 ball at   0.00000,   0.00000  75.00000 deg  75.00000 ips
DIAGNOSTIC: 2 balls allocated
DIAGNOSTIC: 1 nodes allocated.
Launched  Yellow 9A 3 ball at   0.00000,   0.00000 at  19.41143,  72.44444
Loaded    Blue 23 4 ball at   0.00000,   0.00000  90.00000 deg  75.00000 ips
ERROR: failed to allocate ball
Loaded Magenta 2C 5 ball at   0.00000,   0.00000 105.00000 deg  75.00000 ips
DIAGNOSTIC: 3 balls allocated
DIAGNOSTIC: 2 nodes allocated.
Launched Magenta AC 5 ball at   0.00000,   0.00000 at -19.41143,  72.44444
Loaded    Cyan 35 6 ball at   0.00000,   0.00000 120.00000 deg  75.00000 ips
DIAGNOSTIC: 4 balls allocated
ERROR: linkedlist.c: Failed to malloc a Node
```

```
ERROR: Unable to insert into in-play list, freeing B5
DIAGNOSTIC: 2 balls freed
Loaded   White 3E 7 ball at   0.00000,   0.00000 135.00000 deg  75.00000 ips
DIAGNOSTIC: 5 balls allocated
DIAGNOSTIC: 3 nodes allocated.
Launched   White BE 7 ball at   0.00000,   0.00000 at -53.03301,  53.03301
Final scanf call returned -1
```

And much later on we get:

```
ERROR: linkedlist.c: Failed to malloc a Node
ERROR: Unable to insert into off table list, freeing BE
```

In both cases above, compare the bold lines in the output to the underlined failures in the table.

The unreliable code with a student list will have 2 balls score at the end. The unreliable code with the library linked list will have 4 balls score at the end.

Your code is required to have these error messages or ones extremely similar to them. If your code uses malloc where these messages say calloc, adjust the message to match your code. If your code has different function names, adjust to suit. Note that the load message comes as soon as scanf gets data and before the code attempts to allocate. Note that the launch message comes only after insert succeeds.

Reference code issues

It's quite possible that the reference code doesn't deal with allocation failures even if it detects them. Be very sure that your code does properly deal with these issues. Check the DIAGNOSTIC numbers at the end to make sure. Check every insert call in the reference code as well as the ball allocation sequence.