

## Output in Lab 3 SP23

### Text Output

Text output extends existing lab 2 text output.

### Contents

Output in Lab 3 SP23.....	1
Text Output.....	1
Preflight Messages.....	1
Tabular output .....	2
New Messages .....	2
Final Output .....	2
Graphical Output.....	3

### Preflight Messages

```
Loaded 35 6 ball at  -2.00000,    1.50000 -90.00000 deg  90.00000 ips
DIAGNOSTIC: 1 balls allocated
DIAGNOSTIC: 1 nodes allocated.
Launched B5 6 ball at  -2.00000,    1.50000 at    0.00000, -90.00000
Loaded 3E 7 ball at   2.00000,    1.50000 -90.00000 deg  90.00000 ips
DIAGNOSTIC: 2 balls allocated
DIAGNOSTIC: 2 nodes allocated.
Launched BE 7 ball at   2.00000,    1.50000 at    0.00000, -90.00000
Loaded 23 4 ball at  -5.00000,    2.00000 -90.00000 deg  65.00000 ips
DIAGNOSTIC: 3 balls allocated
DIAGNOSTIC: 3 nodes allocated.
Launched A3 4 ball at  -5.00000,    2.00000 at    0.00000, -65.00000
Loaded 2C 5 ball at   5.00000,    2.00000 -90.00000 deg  65.00000 ips
DIAGNOSTIC: 4 balls allocated
DIAGNOSTIC: 4 nodes allocated.
Launched AC 5 ball at   5.00000,    2.00000 at    0.00000, -65.00000
Final scanf call returned -1
```

The above text is the initial output from the file xflip3.output. There are two items highlighted. The yellow highlight is a message that your dynamic memory code prints upon successful allocation of a ball structure. The green highlighted node message comes from the linked list library as part of an insert call. You don't have to deal with node messages in lab 3, other than to make sure that you pass TEXT to the text parameter of insert and deleteSome.

The remaining messages are unchanged from lab 2.

## Tabular output

Tabular output begins and ends with a newline to separate it from everything else.

ST	X	Y	VX	VY	ET=	Score=
A3	-5.00000	2.00000	0.00000	-65.00000	0.000000	0
AC	5.00000	2.00000	0.00000	-65.00000		
B5	-2.00000	1.50000	0.00000	-90.00000		
BE	2.00000	1.50000	0.00000	-90.00000		

The header changed. On the end of the existing header is the score. The rows use the same printf as lab 2.

## New Messages

These are from early on in xflip3.output:

Left flipper:	A3	-5.00000	0.97863	0.00000	-65.73520
Right flipper:	AC	5.00000	0.97863	0.00000	-65.73520
Left flipper:	B5	-2.00000	0.08801	0.00000	-90.73520
Right flipper:	BE	2.00000	0.08801	0.00000	-90.73520

These messages are based on the message code from lab 2 for the wall messages. They show the data before the flipper changes the data.

Wall messages are still required and are unchanged.

## Final Output

When the last ball leaves the table it is time for final output:

```
Off table: AC      1.40000 -0.25377      -6.40000  -17.20225
DIAGNOSTIC: 8 nodes allocated.
17 points
DIAGNOSTIC: 4 nodes freed.
```

The final scores was 72 points:

A3	17 points
AC	17 points
B5	19 points
BE	19 points

Deleted 0 balls from in play list.

DIAGNOSTIC: 1 balls freed

DIAGNOSTIC: 5 nodes freed.

DIAGNOSTIC: 2 balls freed

DIAGNOSTIC: 6 nodes freed.

DIAGNOSTIC: 3 balls freed

DIAGNOSTIC: 7 nodes freed.

DIAGNOSTIC: 4 balls freed

DIAGNOSTIC: 8 nodes freed.

Deleted 4 balls from off table list.

Total runtime is 0.001683950 seconds

The top section starts with the last off-table message that you are used to from lab 2. At the point in time where the ball being removed from the in-play list is successfully inserted onto the off-table list, generate a score message. In the output above, it was for 17 points. The diagnostic messages come from the linked list and your code does not generate them in lab 3.

The middle section of tabular output is generated by printing a message with the final score and then calling `iterate` on the off-table list, asking it to print the score that the ball generated.

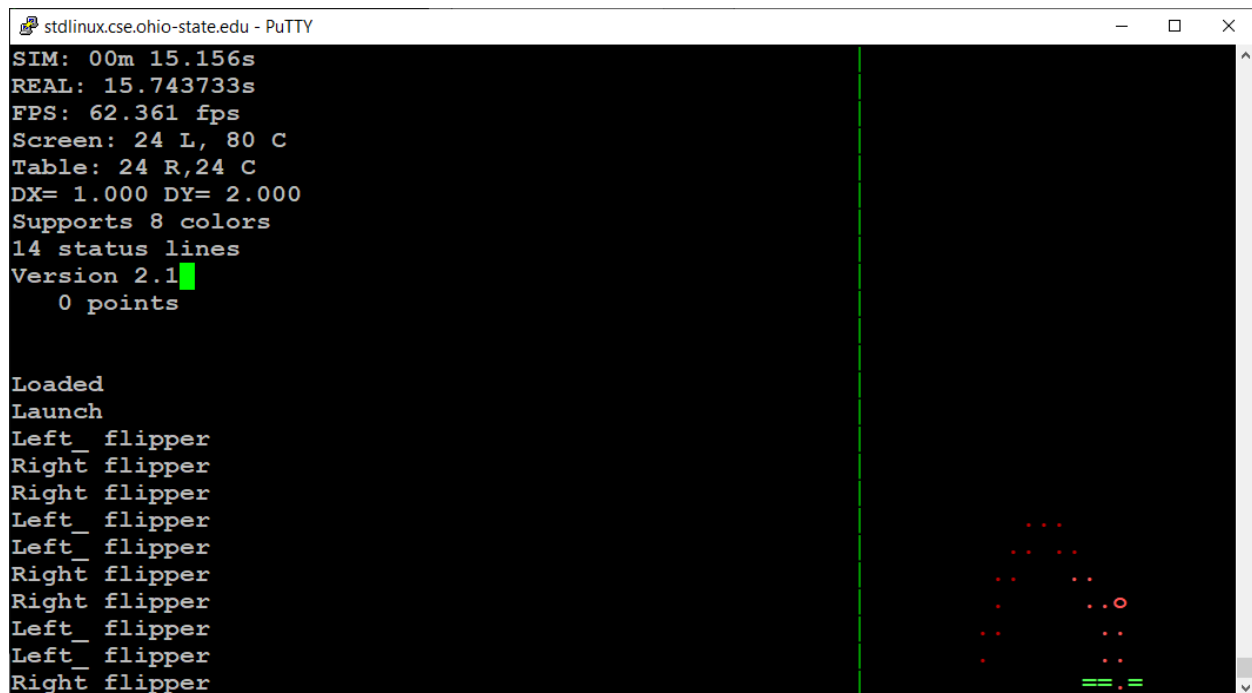
The last section of messages is part of clearing the two lists. The `deleteSome` function returns a value. It should be zero for the in-play list and so deleting from that list should not generate any diagnostics. The diagnostics shown are from clearing the off-table list. The yellow highlight shows a diagnostic generated by your lab 3 dynamic memory code. The green highlight comes from the linked list and you don't have to generate those messages in lab 3. You do have to pass `TEXT` to the `text` parameter of `deleteSome`. You have to print the value returned from `deleteSome` as well. The total runtime message is generated in `main` the same way as in lab 2.

## Graphical Output

Graphics are **required** in lab 3. Feel free to graze through the lab 2 reference code for the various calls and how they are sequenced. The lab 2 graphics write up is certainly worth a look. All of what follows assumes that the code is in graphics mode. None of this happens in text mode.

*You need version 2.1 or higher of `libpb.a` in order to complete lab 3. Note the green cursor in the output below.*

In the following screenshot, the green = are the result of calls to `pb_left` or `pb_right` that your code will make when a flipper constraint activates.



```
stdlinux.cse.ohio-state.edu - PuTTY
SIM: 00m 15.156s
REAL: 15.743733s
FPS: 62.361 fps
Screen: 24 L, 80 C
Table: 24 R, 24 C
DX= 1.000 DY= 2.000
Supports 8 colors
14 status lines
Version 2.1
  0 points

Loaded
Launch
Left flipper
Right flipper
Right flipper
Left flipper
Left flipper
Right flipper
Right flipper
Left flipper
Left flipper
Right flipper
```

The screenshot shows a PuTTY terminal window with a black background. On the left, white text displays simulation statistics: SIM time (00m 15.156s), REAL time (15.743733s), FPS (62.361), screen resolution (24 L, 80 C), table size (24 R, 24 C), DX (1.000), DY (2.000), supported colors (8), status lines (14), version (2.1), and points (0). Below this, a list of events is shown: Loaded, Launch, and a sequence of flipper actions (Left flipper, Right flipper, etc.). On the right, a green vertical line separates the text from a game state visualization. This visualization shows a pool table layout with red dots representing balls and a red circle representing the cue ball. The cue ball is positioned near the bottom right, and several red dots are scattered across the table area.

Note also that your code is required to call `pb_score` each frame to post the current score.

The messages in the status display on the lower left are the result of calls to `pb_status` when something happens (load, launch, either flipper, any of the walls, off table).

At the top left is the sim clock showing simulated elapsed time. When this screenshot was taken the run was 15 seconds and change. The `xinf.pdb` file will run 40+ seconds. Aside from the last 4 seconds, the sim clock and the real time clock should be close in graphics mode. The library handles showing the real time clock, you don't have to do anything for it.

The example above shows a FPS of 62.519, which is close to the desired 64. Your number should be something less than 64 but close to 64.