CMSC216: Bonus Review 1B

Chris Kauffman

Last Updated:
Mon Sep 29 05:32:57 PM EDT 2025

Bonus Review Rules

- 3 Questions will be shown with about 5min per question, 15min total, time limit enforced on Gradescope Quiz
- Individual student bonus dots will be calculated as

```
BonusDots = floor(log2(TotalCorrectSectionAnswers))
            - YourIncorrectAnswers
```

- Cooperation is allowed and encouraged within your discussion section: the more correct answers in the section, the more bonus points for all
- Staff will try to facilitate discussion but will not comment on correct/incorrect answers during the quiz
- Scores will posted after all sections have taken the done the bonus review, likely the following day
- Student in the Discussion Section with the highest `TotalCorrectSectionAnswers` will get +2 BonusDots
- Bonus Review is Open Resource just like the exam: https://www.cs.umd.edu/~profk/216/exam-rules.pdf

Staging

- Open up the Gradescope Bonus Review Quiz for the day
- Once started, the quiz closes after 15min
- Get your resources set for the quiz

Okay...

Question 1

Problem 4 of P1 implemented the `stock_multiplot()` function with this prototype:

```
void stock_multiplot(stock_t **stocks, int stocks_count,
                     int max_height);
```

Which best describes how to access the maximum price in the 4th stock being plotted in this function body?

- (A) `stocks[4]->prices[stocks[4].hi_index]`
- (B) `stocks[4].prices[stocks[4]->hi_index]`
- (C) `stocks[4]->prices[stocks[4]->hi_index]`
- (D) `stocks[4].prices[stocks[4].hi_index]`
- (E) `stocks->prices[stocks[4][hi_index]]`
- (F) `stocks[4][stocks.hi_index].prices`

Question 2

Study to code below and the associated Valgrind output.

```
1  typedef struct {
2    float *data; int count; float total;
3  } samp_t;
4
5  void samp_normalize(samp_t *samp){
6    for(int i=0; i<samp->count; i++){
7      samp->data[i] /= samp->total;
8    }
9  }
10
11 int main(){
12   ...
13 }
```

```
>> gcc -g sample.c
>> valgrind ./a.out
==973== Invalid read of size 4
==973==    at 0x4116F: samp_normalize (sample.c:6)
==973==    by 0x41196: main (sample.c:12)
==973== Address 0x8 is not stackd, mallocd or
==973==    (recently) freed
==973==
==973== Process terminating with default action of
==973==    signal 11 (SIGSEGV):
```

Which of the following is likely contributing to the bug?

- (A) Likely the `data` field is `NULL`
- (B) Likely the `data` field has too little memory allocated
- (C) Likely the `count` field is uninitialized
- (D) Likely the `total` field has too little memory allocated
- (E) Likely the `samp` variable is `NULL`
- (F) Likely the `i` variable is `NULL`

Question 3

The following bit sequence appears in memory at address #4256.

`#4256: 01001111011010100111101100100100`

Which of the following best describes this sequence?

- (A) It is the signed integer `1332378404`
- (B) It is a pointer to memory address `0x4F6A7B24`
- (C) It is the floating point value `3.933939e+09`
- (D) It is the string `${jO`
- (E) It is an uninitialized heap blockk
- (F) It is impossible to say based on the given information