1) (5 pts) ALG (Dynamic Memory Management in C)

Consider the following code that attempts to allocate a new array with the same number of rows as arr, but triple the columns, copy the values from arr into the new array (in the same slots), free the dynamically allocated memory pointed to by arr and return a pointer to the new dynamically allocated array (newarr):

```
int ** tripleCols(int ** arr, int rows, int cols)
{
    int ** newarr = malloc(sizeof(int *) * rows * 3); // point a

    for(int r = 0; r < rows; ++r)
    {
        newarr[r] = malloc(sizeof(int) * cols * 3);
        newarr[r] = arr[r]; // point b
        free(arr[r]);
    }

    free(arr);
    return newarr;
}
```

The key errors in the code are at points a and b, noted in the comments.

(a) (1 pt) What is the fix for the line of code for point a? (This one's simple, so no need to write out the new line of code, just describe the fix.)

Remove the "* 3" since the number of rows needs to be the same.

(b) (2 pts) Why is the line of code for point b incorrect conceptually?

This line of code reassigns a pointer (newarr[r]) to point to a different location instead of copying over the values from arr to newarr.

(c) (2 pts) Write two lines of code to replace this one line of code so that the function will work as planned.

```
for (int c=0; c<cols; c++)
    newarr[r][c] = arr[r][c];
```

Grading: part (a) is all or nothing, 1 pt, award 0, 1 or 2 pts for part (b) as you see fit, for part (c), give full credit if it's correct, 1 pt if there's some loop but it's not fully correct, or 0 if there's no loop.