CS 2110 — Quiz 6 (30 minutes)
November 14th, 2018

Full name: ______________________

GT username: ______________________

This quiz is worth a total of **100 points**.

In accordance with the Georgia Institute of Technology Honor Code, I have neither given nor received aid on this quiz.

Signature: ______________________

Please make sure all of your answers are contained within the answer boxes or the fill-in lines. You have been provided with scratch paper for your work. You will **NOT** be given credit for showing work. Having anything except the answer inside the boxes or above the fill-in lines might cause incorrect results. **Write your name and answers legibly. You will not receive credit for illegible answers.**

Warning: All code you write MUST compile with the standard homework flags:

```
-std=c99 -pedantic -Wall -Werror -Wextra
```

What's the Point?

1. Consider the following code segment:

```
char      a[5][10];
short     b[25];
int       c[20];
struct s  d[10][20][30];
```

Using pointer arithmetic complete the following expressions. **You may not use `[` or `]`!**

(a) Extract the seventh `short` in `b`: 5

`short shortValue =` *(b + 6) `;`

(b) Find the address of the second `int` in `c`: 5

`int     *intAddr =` (c + 1) `;`

(c) Extract the `char` at `a[3][2]`: 7

`char   charValue =` *(*(a + 3) + 2) `;`

(d) Find the address of the `struct s` at `d[5][7][10]`: 7

```
struct s *pd = &d[0][0][0];
struct s *structAddr = pd + 5 * (20 * 30) + 7 * (30) + 10;
```

Searching for a Book

2. The function `findBestBook` has three parameters: `books` (an array of `Book`s), `size` (the number of `Book`s in `books`) and `bookComp` (a user-supplied function for comparing two `Book`s). 16

Complete the function definition by filling in the correct parameter type for `bookComp`.

```

Book *findBestBook(Book *books, int size, Book* (*bookComp)(Book*, Book*))
{
  if ((!books) || (!bookComp)) return NULL;

  Book *bestBook = &books[0];

  for (int i = 1; i < size; i++)
    bestBook = (*bookComp)(bestBook, &books[i]);

  return bestBook;
}
```

Extracting Channels of a Pixel

3. Write a function `extractChannels` which takes a `u32 pixel` (see diagram below) and returns the three color channels through `u32*` parameters `red`, `green`, and `blue`. Each color channel consists of 10 bits and the uppermost bits, `[31:30]`, are unused. *Note:* `u32` is an alias for `unsigned int` on ARM. 20

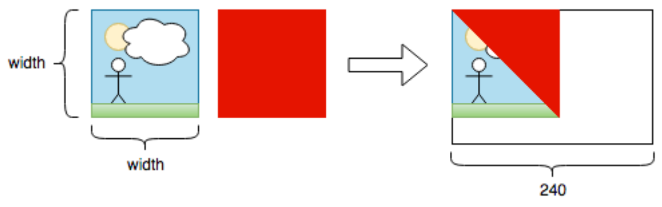
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
		G	G	G	G	G	G	G	G	G	G	R	R	R	R	R	R	R	R	R	R	B	B	B	B	B	B	B	B	B	B

Reminder: The color channel parameters are pointers to 32-bit values!

```
void extractChannels(u32 pixel, u32 *red, u32 *green, u32 *blue)
{
    *green = (pixel >> 20) & 0x3FF;
    *red   = (pixel >> 10) & 0x3FF;
    *blue  = (pixel      ) & 0x3FF;
}
```

Drawing a Collage with DMA

4. The function `drawSquareDiagonalCollage` collages a square `image` and a `color` along the `image`'s diagonal: Row zero consists of one pixel of the `image` and the remainder of the `color`. The final row consists entirely of the `image`. 40



Do **not** copy the full `image` or a full square of the `color`, only the portions appearing in the collage.

Note: You're allowed to call DMA for small copies.

```
#define GBA_HEIGHT 160
#define GBA_WIDTH  240

#define OFFSET(r, c, w) (((r) * (w)) + (c))

#define DMA_DST_INC  (0 << 21)
#define DMA_DST_DEC  (1 << 21)
#define DMA_DST_FIX  (2 << 21)
#define DMA_DST_RST  (3 << 21)

#define DMA_SRC_INC  (0 << 23)
#define DMA_SRC_DEC  (1 << 23)
#define DMA_SRC_FIX  (2 << 23)

#define DMA_ON       (1 << 31)
```

```
volatile unsigned short *videoBuffer = (unsigned short *) 0x6000000;

void drawSquareDiagonalMontage(const u16 *image, int width, u16 color)
{
  for (int row = 0; row < width ; row++)
  {
    DMA[3].src = image + OFFSET(row, 0, width) ;  // Draw the image portion

    DMA[3].dst = videoBuffer + OFFSET(row, 0, GBA_WIDTH) ;

    DMA[3].cnt = (row + 1) | DMA_ON | DMA_SRC_INC | DMA_DST_INC

                 __________________________________;  // Continue DMA[3].cnt here

    DMA[3].src = &color ;  // Draw the rectangle portion

    DMA[3].dst = videoBuffer + OFFSET(row, row + 1, GBA_WIDTH) ;

    DMA[3].cnt = (width - (row + 1)) | DMA_ON | DMA_SRC_FIX | DMA_DST_INC

                 __________________________________;  // Continue DMA[3].cnt here
  }
}
```