CMPU-224 Lab5 Practice Solutions
Spring 2025

1. Consider the following structure declaration:

```
struct question {
    short  a;
    int    b;
    char   c;
    double d;
};
```

   (a) Give the total size of the structure in bytes (i.e., the value of the `sizeof(struct question)` expression).

   `sizeof(struct question)` returns **24**

   (b) If the structure declaration above was rearranged to minimize the space of the structure, what would be the new size of the structure?

   Total size of rearranged and minimized `struct` in bytes: **16**

   (c) Assume you have the following C code utilizing the above (unminimized) structure:

```
long prob(struct question *x) {
    return x->b;
}
```

   Fill in the missing blanks below for the assembly language implementation of the above function.

```
prob:
    lw a0, 4(a0)
    ret
```

2. You have the following two-dimensional array declared as follows:

```
int matrix[7][4];
```

   Assume the start of the array is at address `0x4000`.

   What are the indices (`x` and `y` below) of the array that refers to the array element at address `0x4058`?

Solution: Answer: x = 5, y = 2 (`matrix[5][2]`)

Explanation:

1. Find the byte offset: `0x4058` - `0x4000` = `0x58` bytes.
2. Convert hex to decimal: `0x58` in hex is $(5 \times 16) + 8 = 88$ bytes.
3. Find the element offset: 88 bytes / 4 bytes per `int` = 22 elements.
4. Calculate indices based on row length (4 elements per row): $22 \div 4 = 5$ with a remainder of 2.
5. Therefore, row index `x` is 5, and column index `y` is 2.

3. Consider the following C function that accesses an array of 32-bit integers and checks a condition:

```
int check_threshold(int arr[], int index) {
    if (arr[index] >= 50) {
        return 1;
    }
    return 0;
}
```

Below is the corresponding assembly code. Fill in the missing RV32 assembly instructions to make the code function correctly.

```
check_threshold:

  slli t0, a1, 2

  add t0, a0, t0

  lw t1, 0(t0)

  li t2, 50

  blt t1, t2, .END

  li a0, 1

  ret

.END:

  li a0, 0

  ret
```

Solution:

- **Blank 1:** `slli t0, a1, 2` (Multiplies index by 4 to get the byte offset).
- **Blank 2:** `lw t1, 0(t0)` (Loads the 32-bit integer from memory).
- **Blank 3:** `blt t1, t2, .END` (Branches to the false condition if `arr[index] < 50`).