Consider the following C functions and the assembly code they compile to:

```c
int foo(int * a, int b){
    int x = 0;
    int i;
    for(i = b-1; i > -1; i--){
        if(i != 47){
            a[i] = x;
        } else {
            a[i] = 47;
        }
        x += i;
    }
    return x;
}

int main(int argc, char ** argv){
    int a[4];
    int y = foo(a, 4);
    return x;
}
```

1. For each variable, indicate which register that variable is stored in.
   - a: %rdi
   - b: %rsi
   - x: %rax
• i: %rdx

2. Based on the assembly code, fill in the blanks in the C source code. You may only use the C variable names a, b, x, i, not register names. Use array notation to show any accesses to elements of a, not pointer arithmetic.

**Hint:** cmp a, b sets the same condition codes as b-a

3. Below is a diagram of the stack at the beginning of function main (that is, immediately before the instruction subq $24, %rsp is executed). Modify this diagram to show the state of the stack immediately before the function foo returns (that is, immediately before the instruction retq is executed). Include in your diagram an arrow labeled %rsp that indicates the address stored in the register %rsp at that point and an arrow labeled a that indicates the address stored in the variable a at that point.

**Hints:** Remember that you are running on a 64-bit little-endian machine.