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

```c
int foo(int * a, int b) {
    int x = ________;  
    int i;
    for(i = ________; ________; i--) {
        if(__________) {
            __________;
        } else {
            __________;
        }
        __________;
    }
    return ________;
}

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

1. For each variable, indicate which register that variable is stored in.

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

\textbf{Hint:} \texttt{cmp a, b} sets the same condition codes as \texttt{b-a}

3. Below is a diagram of the stack at the beginning of function \texttt{main} (that is, immediately before the instruction \texttt{subq $24, %rsp} is executed). Modify this diagram to show the state of the stack immediately before the function \texttt{foo} returns (that is, immediately before the instruction \texttt{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 \texttt{a} that indicates the address stored in the variable \texttt{a} at that point.

\textbf{Hints:} Remember that you are running on a 64-bit little-endian machine.