Exercise: Stack Frames for Function Invocations

The stack frame for a function invocation contains arguments (in reverse order), return address, frame pointer of caller, and local variables. Remember the stack grows towards the low address.

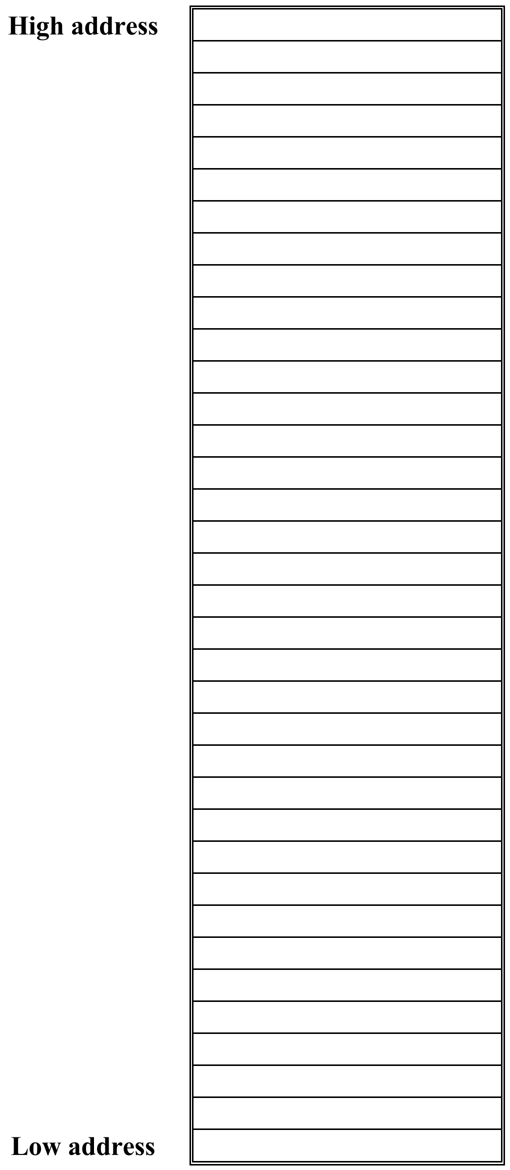
```
int main() {
   int p = 5, q = 10;
   printf("%d\n",func(p,q));
}
```

```
void func(int a, int b) {
   int c = a + b;
    return c;
}
```

An `int` type requires 4 bytes of memory. Return address (RA) and previous frame pointer (PFP) fields take 4 bytes each. Assume that the storage for local variables is allocated in the order of declaration in the source code and the stack protector/guard is not enabled. Assume the program is run on a little-endian machine.

Show the stack frames that you will find on the stack when the program is currently executing inside the func(). **Show where the current frame pointer during the `func()` invocation points at in the stack frame.**

Important: each cell in the stack segment memory block below represents one byte (8 bits) of memory.


High address
Low address