```
class CacheController:
    def __init__(self):
        # Initialize all caches to INVALID state
        self.caches = ['I', 'I', 'I', 'I']  # C1, C2, C3, C4

    def process_request(self, a, b, c):
        """
        Process a cache request according to MESI protocol.
        a: processor number (1-4)
        b: operation type (R for read, W for write)
        c: continue flag (0 to terminate, 1 to continue)
        """
        processor = a - 1  # Convert to 0-based index
        operation = b

        # Apply MESI protocol rules
        if operation == 'R':
            self._handle_read(processor)
        elif operation == 'W':
            self._handle_write(processor)

        # Return current cache states and continue flag
        return self.caches, c == 1

    def _handle_read(self, processor):
        """Handle a read operation from the specified processor."""
        # Check current state of the processor's cache
        current_state = self.caches[processor]

        if current_state == 'I':
            # If any other cache has the line in M or E state, it
changes to S
            # and the requesting cache gets S
            if 'M' in self.caches or 'E' in self.caches:
                for i in range(4):
                    if self.caches[i] == 'M' or self.caches[i] == 'E':
                        self.caches[i] = 'S'
                self.caches[processor] = 'S'
            # If any other cache has the line in S state, requesting
cache gets S
            elif 'S' in self.caches:
                self.caches[processor] = 'S'
            # If no other cache has the line, requesting cache gets E
            else:
                self.caches[processor] = 'E'
        # If the state is already M, E, or S, it remains unchanged

    def _handle_write(self, processor):
        """Handle a write operation from the specified processor."""
        # For a write operation, all other caches must invalidate
```

```
their copies
        for i in range(4):
            if i != processor:
                self.caches[i] = 'I'

        # The writing cache gets the M state
        self.caches[processor] = 'M'

def main():
    controller = CacheController()
    continue_processing = True

    while continue_processing:
        # Get input from the user
        try:
            a, b, c = input("Enter request (a,b,c): ").replace('(',
'').replace(')', '').split(',')
            a = int(a)
            c = int(c)

            if a < 1 or a > 4 or b not in ['R', 'W'] or c not in [0,
1]:
                print("Invalid input. Please use format (a,b,c)
where:")
                print("a is 1-4, b is R or W, c is 0 or 1")
                continue

            # Process the request
            cache_states, continue_processing =
controller.process_request(a, b, c)

            # Display the result
            print(f"Cache states: C1={cache_states[0]},
C2={cache_states[1]}, C3={cache_states[2]}, C4={cache_states[3]}")

        except ValueError:
            print("Invalid input format. Please use format (a,b,c)")
        except Exception as e:
            print(f"Error: {e}")

if __name__ == "__main__":
    main()

Enter request (a,b,c): (1,W,0)
Cache states: C1=M, C2=I, C3=I, C4=I
```