1 Dynamic Control Flow.

1a Consider the following procedures.

```
void m(int (*f)(int, int), int* a, int* b, int *c, int n) {
  for (int i=0; i<n; i++)
    c[i] = f(a[i], b[i]);
}

int r(int (*f)(int, int), int* a, int n, int p) {
  for (int i=0; i<n; i++)
    p = f(a[i], p);
  return p;
}

int fi(int (*f)(int, int), int *a, int *b, int n, int p) {
    int j = 0;
    for (int i = 0; i < n; i++) {
        if (f(a[i], p)) {
            b[j++] = a[i];
        }
    }
    return j;
}
```

Starting with two arrays of `n` positive integers, `a` and `b`, compute the following values. Your solution must not directly call any procedures other than `m` , `r` or `fi`. However, you may define your own functions to pass as parameters. Your code must not contain any loops. Your solution may call any of the procedures multiple times, and may include basic math operations on individual values, and may allocate to the heap as needed.

(i) Allocate and compute a new array, `c`, that stores the sum of elements from `a` and `b`; e.g., if `a=[1,2]` and `b=[2,3]`, the new array is `c=[3,5]`. Assume that all of `a`, `b` and `n` are defined.

(ii) Determine the maximum value in the array `c` computed in Part i and store it in a variable called `mx`.

(iii) Allocate and compute two new arrays, `e` and `o`. And store the even elements of 'a' in `e` and the odd elements of b in 'o'. Store their lengtss in `n_even` and `n_odd` respectively

(iv) Determine the minimum of all the elements in ‘e‘ and ‘o‘ and store it in a variable called `mn`. Assume both ‘e‘ and ‘o‘ have at least 1 element.

(v) Count the number of elements of `c` that are in between `mx` and `mn`