1 **Dynamic Allocation.** Consider each of the following pieces of C code to determine whether it contains (or may contain) a memory-related problem. Label each of the following code snippets as one of the following:

A: There is no memory leak or dangling pointer; nothing needs to be changed with malloc or free.
B: There is no memory leak or dangling pointer, but the code would be improved by moving malloc or free.
C: There is a possible memory leak that is best resolved by adding, removing or moving malloc or free.
D: There is a possible dangling pointer that is best resolved by adding, removing or moving malloc or free.
You can assume that the starting point for each snippet of code is a call to `foo`, and that `copy` is in a different module. Do not fix any bugs; for each part, fill in a single multiple choice bubble based off of the options above. Most of the code snippets are very similar. Changes from previous versions and/or key things to look for in **bold** font.

1a

```
int* copy(int s) {
    int* d = malloc(sizeof(int));
    *d = s;
    return d;
}
```

```
void foo (int s) {
    int* d = copy(s);
    printf("value is %d", *d);
    free(d);
}
```

A B C D

1b

```
int* copy(int s) {
    int* d = malloc(sizeof(int));
    *d = s;
    free(d);
    return d;
}
```

```
void foo (int s) {
    int* d = copy(s);
    printf("value is %d", *d);
}
```

A B C D

1c

```
void copy(int s, int* d) {
    *d = s;
}
```

```
void foo (int s) {
    int d = 0;
    copy(s, &d);
    printf("value is %d", d);
}
```

A B C D

1d

```
void copy(int s, int* d) {
    *d = s;
}
```

```
void foo (int s) {
    int* d = malloc (sizeof(int));
    copy(s, d);
    free(d);
    printf("value is %d", *d);
}
```

A B C D