What is the value printed under the following schemes (assume left-to-right evaluation):

a. Call-By-Value?
b. Call-By-Reference?
c. Call-By-Value-Result?
d. Call-By-Name?
e. Call-By-Need?

```c
int x;
int f(int y, int z, int zz) {
    int w = x + 2;
    z++;
    zz++;
    return y + y + z + w;
}
int main() {
    x = 1;
    printf("%i", f(++x, x, x) + x);
}
```

d) Call-By-Name
In call-by-name, instead of evaluating the arguments(including ++x) before passing them to f, the arguments y, z and zz become macros for ++x, x and x, respectively. Thus, z++ and zz++ become x++, and y + y + z + w becomes ++x + ++x + x + w.
So any change made to x (through the macros) will be reflected in main().

```c
int x;
int f(int y, int z, int zz) {
    int w = x + 2;                  //w = 1+2 =3
    z++;                            //x++ (x is now 2)
    zz++;                           //x++ (x is now 3)
    return y + y + z + w;     //return (++x)+(++x)+(x)+w
                             //x is 4 -> 5 ->5
                             //So return value is 4+5+5+3 =17
}
int main() {
    x = 1;
    printf("%i", f(++x, x, x) + x);  //x is still 1 at fn call
    //After function
    //returns, x is now 5.
    //So output is 17+5 = 22
```
is stored and fetched if z is used again. Similarly, for zz and y.

```c
int x;
int f(int y, int z, int zz) {
    int w = x + 2; // w = 1 + 2 = 3
    z++; // x++ (x is now 2, z’s value is stored as 2)
    zz++; // x++ (x is now 3, zz’s value is stored as 3)
    return y + y + z + w; // First time y is computed as 4 and saved.
}                                // Function returns 4 + 4 + 2 + 3 = 13

int main() {
    x = 1;
    printf("%i", f(++x, x, x) + x); // x is still 1 at fn call
}                                    // After function returns, x is now 4.
                                          // So output is 13 + 4 = 17
```