```
#include<stdio.h>
#include<math.h>

double bad_limit_calc() {
    float previous = 100;
    for (int i = 2; i<71 ;i++) {
        float x_mult = 100*previous;
        float x_div = x_mult/i;
        previous = x_div;
        printf("i = %i, an = %f \n", i+1, previous);
    }
    return previous;
}

double factorial_calc(int i) {
    if (i == 0){
        return 1;
    }
    return i * factorial_calc(i-1);
}

double good_limit_calc() {
    double previous = 100;
    for (int i = 2; i<300 ;i++) {
        double x_mult = 100*previous;
        double x_div = x_mult/i;
        previous = x_div;
        printf("i = %i, an = %f \n", i+1, previous);
    }
    return previous;
}

void main(){
    printf("%f \n", bad_limit_calc());
    printf("%f \n", good_limit_calc());
}
```

Explanation:

The issue with the student's program is that at n = 56, the value of the numerator becomes higher than the upper limit of the IEEE double format (2^{53}), meaning it becomes "inf". However, this is not accurate to the limit, because over time, the factorial in the denominator will become larger than the numerator, bringing the limit to 0. The use of double precisions means that we can calculate the limit up to its peak (n=100 or 101), and then watch it come back down.