

hw7 az147

1.

A computer program execution having the presence of aging factors means that the longer the program runs, the partial failures will accumulate and more likely to cause errors.

An example is that one old version of browser which has memory leak bugs underlying. Longer the time browser is working, it will take up more memory space. Finally the browser take up too much memory and slow down the computer severely, in which case we would have to restart the browser or the computer.

2.

2.a

the most-parent process will never quit and keep spawning new processes and not controlling the number of children processes.

2.b

EAGAIN error, when the system-imposed limit on the number of processes was encountered.

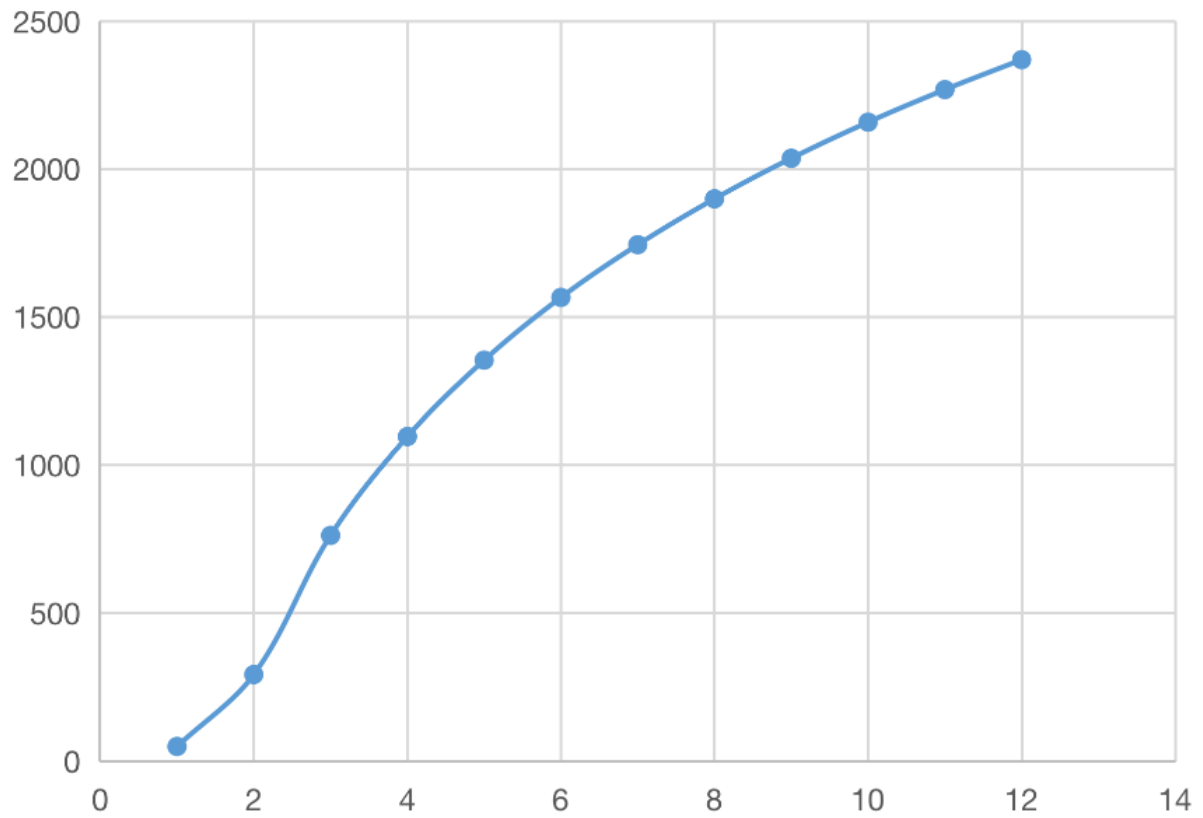
2.c

the most-parent process keep spawning new processes and the number of processes may hit the limit overtime. and the program can not process any new requests.

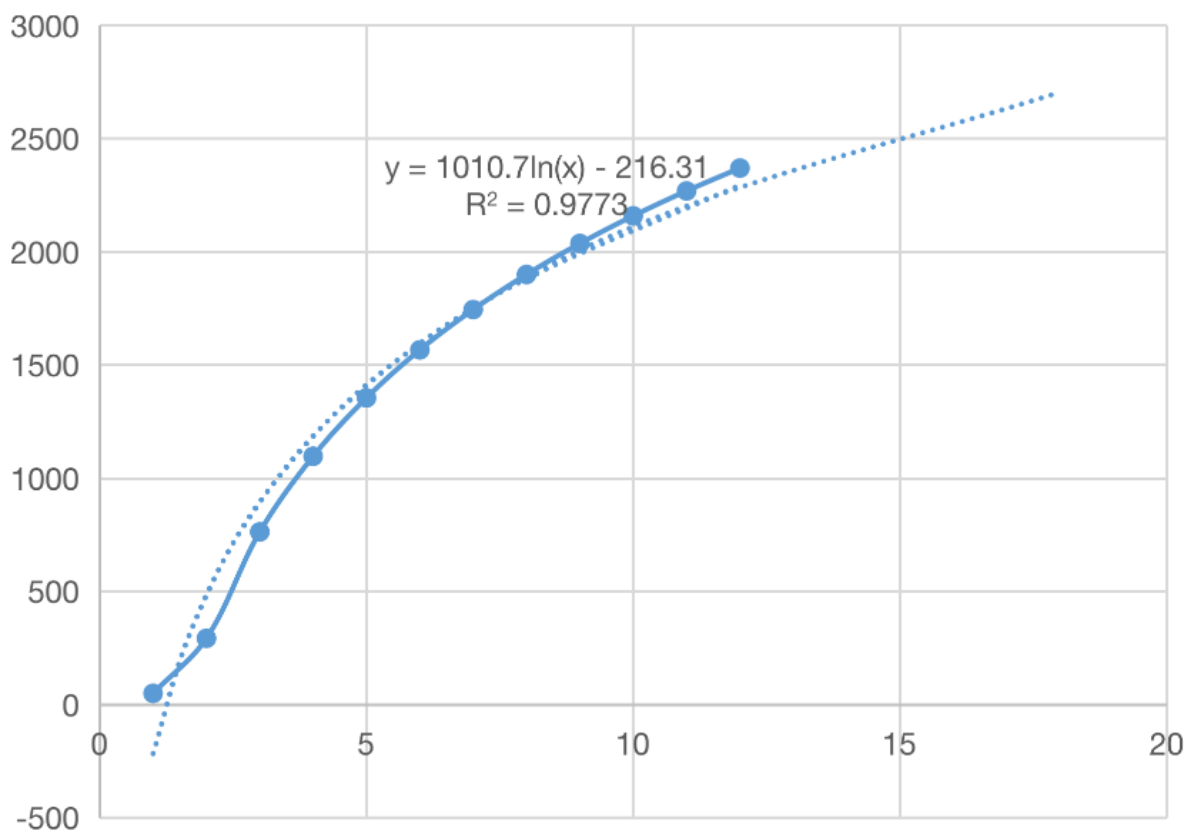
3.

3.a

the aging effect accumulation rate is non-linear, (it is \ln)



2.8 gigabytes = 2800 megabytes



according to the simulation function, the time to AR-failure is $19.7742 = 20$ months.

Extra Credit:

- a. No, because the program is not thread-safe, it will have data racing between thread while incrementing counter. and the counter we get will be less than 100 after finishing the program.
- b. Yes, it is an AR-fault. Because as the num_threads goes up, the longer the program runs, we will get larger deviation between actual counter and expected answer.
- c.

```
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#define N_THREADS 100
int counter;
void *thread_code (void *arg) {
    counter = counter + 1;
    pthread_exit(0);
}
void start_threads() {
    int i;
    pthread_t threads[N_THREADS];
    counter = 0;
    for (i = 0; i < N_THREADS; i++) {
        pthread_create(&threads[i], NULL, thread_code, NULL);
    }
    for (i = 0; i < N_THREADS; i++) {
        pthread_join(threads[i], NULL);
    }
    printf("Counter value: %d\n", counter);
}
void test(int nums){
    int pass = 0;
    for(int i = 0; i < nums; i++) {
        counter = 0;
        start_threads();
        if(counter == N_THREADS) {
            pass++;
            printf("test %d out of %d PASSED\n", i + 1, nums);
        } else {
            printf("test %d out of %d FAILED\n", i + 1, nums);
        }
    }
    printf("passed %d tests out of %d tests\n", pass, nums);
}
int main() {
    test(100);
}
```

```
    return EXIT_SUCCESS;  
}
```

for 100 tests:(pass rate 99%)

```
test 88 out of 100 PASSED  
Counter value: 100  
test 89 out of 100 PASSED  
Counter value: 100  
test 90 out of 100 PASSED  
Counter value: 100  
test 91 out of 100 PASSED  
Counter value: 100  
test 92 out of 100 PASSED  
Counter value: 100  
test 93 out of 100 PASSED  
Counter value: 100  
test 94 out of 100 PASSED  
Counter value: 100  
test 95 out of 100 PASSED  
Counter value: 100  
test 96 out of 100 PASSED  
Counter value: 100  
test 97 out of 100 PASSED  
Counter value: 100  
test 98 out of 100 PASSED  
Counter value: 100  
test 99 out of 100 PASSED  
Counter value: 100  
test 100 out of 100 PASSED  
passed 99 tests out of 100 tests
```

for 1000 tests:(pass rate 97.7%)

```
test 990 out of 1000 PASSED  
Counter value: 100  
test 991 out of 1000 PASSED  
Counter value: 100  
test 992 out of 1000 PASSED  
Counter value: 100  
test 993 out of 1000 PASSED  
Counter value: 100  
test 994 out of 1000 PASSED  
Counter value: 100  
test 995 out of 1000 PASSED  
Counter value: 100  
test 996 out of 1000 PASSED  
Counter value: 100  
test 997 out of 1000 PASSED  
Counter value: 100  
test 998 out of 1000 PASSED  
Counter value: 100  
test 999 out of 1000 PASSED  
Counter value: 100  
test 1000 out of 1000 PASSED  
passed 977 tests out of 1000 tests
```

for 10000 tests:(pass rate 98.45%)

```
test 9990 out of 10000 PASSED  
Counter value: 100  
test 9991 out of 10000 PASSED  
Counter value: 100  
test 9992 out of 10000 PASSED  
Counter value: 100  
test 9993 out of 10000 PASSED  
Counter value: 100  
test 9994 out of 10000 PASSED  
Counter value: 100  
test 9995 out of 10000 PASSED  
Counter value: 100  
test 9996 out of 10000 PASSED  
Counter value: 100  
test 9997 out of 10000 PASSED  
Counter value: 100  
test 9998 out of 10000 PASSED  
Counter value: 100  
test 9999 out of 10000 PASSED  
Counter value: 100  
test 10000 out of 10000 PASSED  
passed 9845 tests out of 10000 tests
```

as the number of tests goes up, the number of fail cases are growing larger, and the number is so large that it is unacceptable. So I will return the program to the test team.