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.

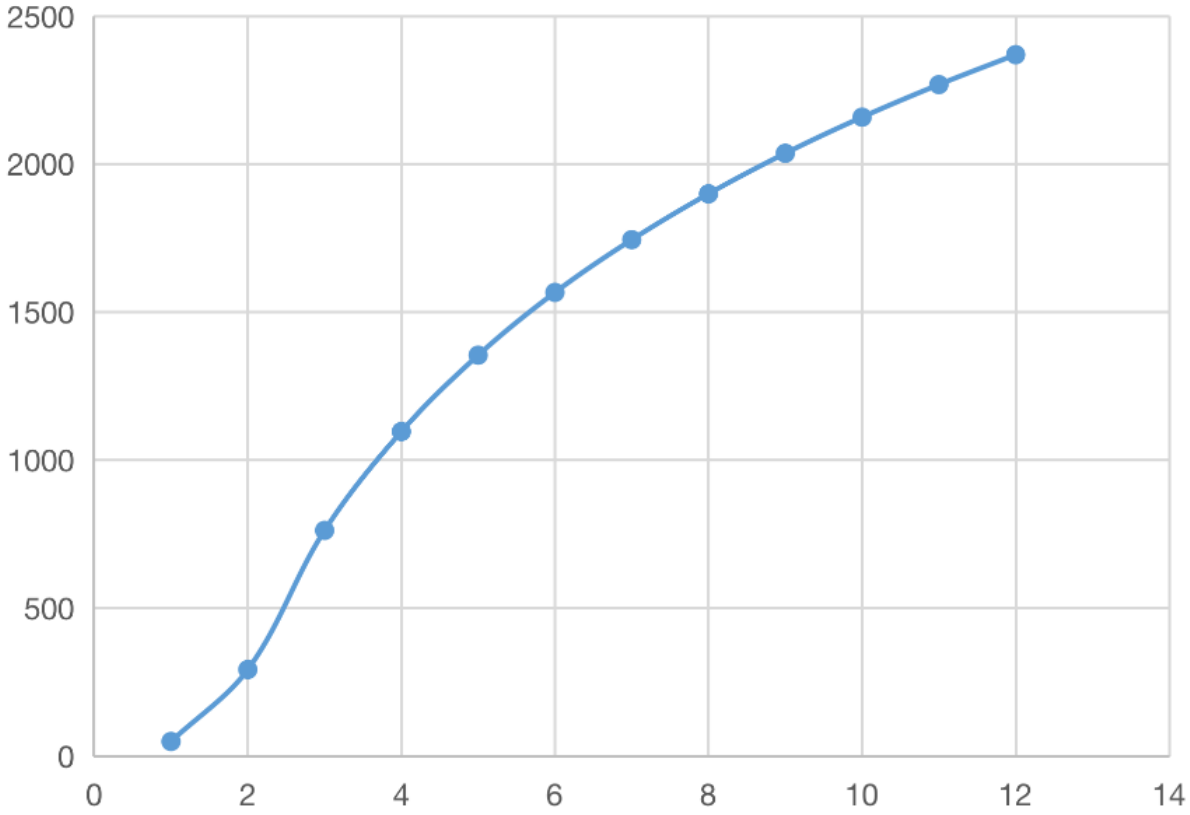
2.c

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.

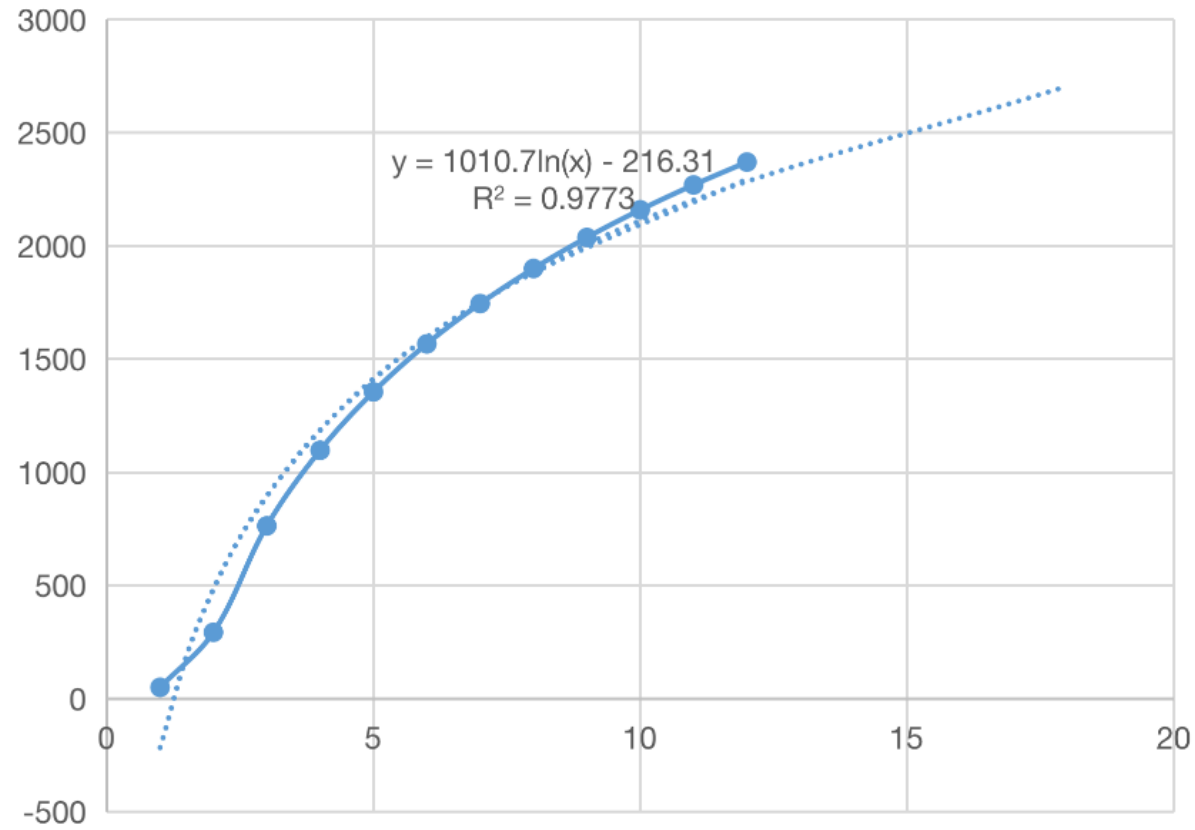
3.

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.

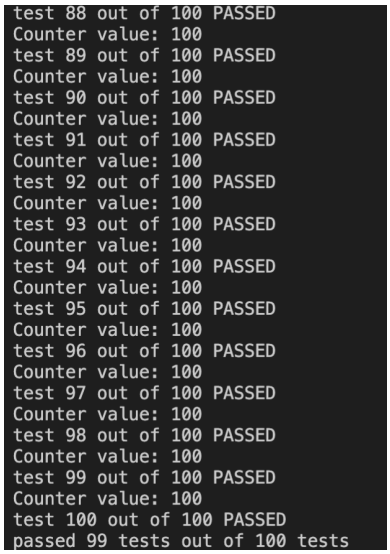
b. Yes, it is an AR-fault. Because as the num_threads goes up, the longer the program runs, we will get larger deviation between actual counter and expected answer.

c.

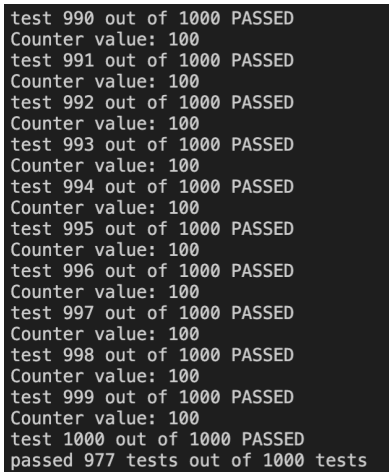
```
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#define N_THREADS 100
int counter;
void *thread_code (void *arg) {
    counter = counter + 1;
    pthread_exit(0);
}
void start_threads() {
    int i;
    pthread_t threads[N_THREADS];
    counter = 0;
    for (i = 0; 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;
}
```

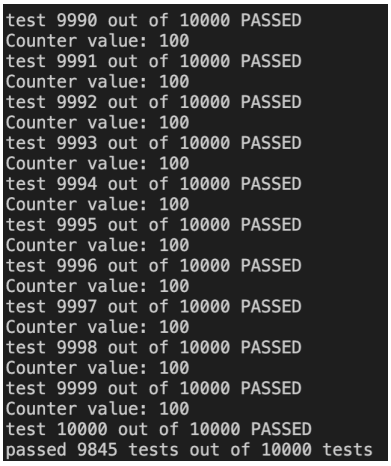
for 100 tests:(pass rate 99%)



for 1000 tests:(pass rate 97.7%)



for 10000 tests:(pass rate 98.45%)



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.