1. In the lecture on fundamentals of software aging, we saw that an AR-fault (ARB) activation occurs under the presence of aging factors. Describe in your own words, what does it mean for a computer program execution to have the presence of aging factors? Give one real or hypothetical (but realistic) example.

2. The C language code listed below contains a fault that is classified as an AR-fault (ARB). Based on this code describe the
   a) AR-fault.
   b) AR-error caused by the activation of the AR-fault identified in (a).
   c) Aging effect caused by the causality chain composed of (a) and (b).

(You may refer to the Linux fork(2) man page for some information https://man7.org/linux/man-pages/man2/fork.2.html)

```c
#include <sys/types.h>
#include <unistd.h>
#include <stdlib.h>

void child_code(void){
    /* Performs the child task. */
    exit();
}

int main(void){
    pid_t child;
    while(1){
        /* Reads the next user's input.
           Below creates a child to process it.
        */
        child=fork();
        if (child == 0){
            child_code();
        }
        else
        if (child < 0){
            printf("Error: Child not created\n");
        }
    }
}
```

3. The embedded software that controls a military satellite suffers from software aging. This satellite's availability must be no less than six nines (99.9999%), which means a yearly downtime no longer than 31.56 seconds. Because of the accumulation of the aging effects, the size of the satellite's main application process grows progressively. The memory size of this process has been monitored monthly, for one year, and the result follows (in megabyte): 50, 293, 763, 1097, 1355, 1567, 1745, 1900, 2037, 2159, 2269, 2370. It is known that once this process grows beyond 2.8 gigabytes, it will be abruptly forced to terminate by the satellite operating system. Taking into consideration the above-described scenario, answer the following questions:
   a) Based on the monitored data, is the aging effect accumulation rate linear or non-linear? Justify your answer.
   b) What is the expected time to AR-failure (in months)? Justify your answer. (Show how you derive the answer)

Supplemental Question (Extra Credit 20%)

According to the specification for the C code listed below, its correct output is expected to be “Counter Value: 100”. Based on this expected value and the taxonomy we saw in the lecture, clearly keeping in mind the concepts of fault, error, and failure, answer the following questions:

a) Based only on the analysis of the source code, is this program fault-free?
   Justify your answer.
b) If the answer to (a) is No, is there an AR-fault (ARB)?
   Justify your answer.
c) Assume you are responsible for the acceptance testing of this program. Implement the acceptance test, define the number of test executions for this program, and perform it.
   For each execution, mark if the test “passed or failed”.
   Attach your program and a screenshot of your test results in your submission.
d) Based on the results obtained in (c), do you recommend this program to be released and deployed, or to return to the test team?
   Justify your 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);
}

int main() {
    start_threads();
    return EXIT_SUCCESS;
}
```