שם: ליאור דניאל
ת.ז.: 305257347

Question 1

לחץ על אחד מהנושאים להסברים נוספים: objdump, nm ,size

השתמשתי בכלי "nm" על מנת לפענח את מיקום המשתנים.
לאחר כתיבה של הפקודה "man nm" נקבל תיאור של כל הסימבולים על מנת להבין היכן המשתנים ממוקמים.

תיאור סימבולים עיקריים Lower case – משתנים לוקאליים. Upper Case – משתנים חיצוניים.
'b' \ 'B' – משתנים שאינם מאותחלים, מאוחסנים ב-Data Segment.
'd' \ 'D' – משתנים המאוחסנים ב-Stack של ה-Data Section.
't' \ 'T' – המשתנים מאוחסנים ב-Text (Code) Section.

1. globBuf – משתנה גלובאלי לא מאותחל, מאוחסן ב-Uninitialized data segment.
2. primes – משתנה גלובאלי מאותחל, מאוחסן ב-Initialized data segment.
3. square – פונקציה נשמרת ב-Text (code) section, אך כשנקראת מאוחסנת ב-Frame על ה-Stack.
4. result – משתנה לוקאלי של הפונקציה, נשמר באותו Frame שנוצר על ה-Stack עבור הפונקציה.
5. 'result' – return – ערך המוחזר באמצעות רגיסטר.
6. doCalc - פונקציה נשמרת ב-Text (code) section, אך כשנקראת מאוחסנת ב-Frame על ה-Stack.
7. t - משתנה לוקאלי של הפונקציה, נשמר באותו Frame שנוצר על ה-Stack עבור הפונקציה.
8. main - פונקציה נשמרת ב-Text (code) section, אך כשנקראת מאוחסנת ב-Frame על ה-Stack.
9. key - משתנה סטטי מאותחל, מאוחסן ב-Initialized data segment.
10. mbuf – משתנה סטטי לא מאותחל, מאוחסן ב-Uninitialized data segment.
11. p - משתנה לוקאלי של הפונקציה, נשמר באותו Frame שנוצר על ה-Stack עבור הפונקציה.

הערה - עבור משתנים לוקאליים המאותחלים בתוך פונקציות מוקצה זיכרון בתוך ה-Frame שנוצר עבור הפונקציה ב-Stack.

```
lior@DESKTOP-8CP0JPA:/mnt/c/Users/Lior/Desktop/Q1$ nm -n p
                 w _ITM_deregisterTMCloneTable
                 w _ITM_registerTMCloneTable
                 w __cxa_finalize@@GLIBC_2.2.5
                 w __gmon_start__
                 U __libc_start_main@@GLIBC_2.2.5
                 U exit@@GLIBC_2.2.5
                 U printf@@GLIBC_2.2.5
0000000000000528 T _init
0000000000000580 T _start
00000000000005b0 t deregister_tm_clones
00000000000005f0 t register_tm_clones
0000000000000640 t __do_global_dtors_aux
0000000000000680 t frame_dummy
000000000000068a t square
00000000000006a0 t doCalc
0000000000000702 T main
0000000000000730 T __libc_csu_init
00000000000007a0 T __libc_csu_fini
00000000000007a4 T _fini
00000000000007b0 R _IO_stdin_used
00000000000007e4 r __GNU_EH_FRAME_HDR
0000000000000974 r __FRAME_END__
0000000000200db0 t __frame_dummy_init_array_entry
0000000000200db0 t __init_array_start
0000000000200db8 t __do_global_dtors_aux_fini_array_entry
0000000000200db8 t __init_array_end
0000000000200dc0 d _DYNAMIC
0000000000200fb0 d _GLOBAL_OFFSET_TABLE_
0000000000201000 D __data_start
0000000000201000 W data_start
0000000000201008 D __dso_handle
0000000000201010 D primes
0000000000201020 d key.2775
0000000000201024 B __bss_start
0000000000201024 D _edata
0000000000201028 D __TMC_END__
0000000000201040 b completed.7698
0000000000201060 b mbuf.2776
0000000000bc5060 B globBuf
```

```
char globBuf[65536];            /* 1. Uninitialized data segment */
int primes[] = { 2, 3, 5, 7 };  /* 2. Initialized data segment */

static int
square(int x)                   /* 3. Allocated in frame for square() */
{
    int result;                 /* 4. Allocated in frame for square() */

    result = x * x;
    return result;              /* 5. Return value passed via register */
}

static void
doCalc(int val)                 /* 6. Allocated in frame for doCalc() */
{
    printf("The square of %d is %d\n", val, square(val));

    if (val < 1000) {
        int t;                  /* 7. Allocated in frame for doCalc() */

        t = val * val * val;
        printf("The cube of %d is %d\n", val, t);
    }
}

int
main(int argc, char* argv[])    /* Allocated in frame for main() */
{
    static int key = 9973;      /* Initialized data segment */
    static char mbuf[10240000]; /* Uninitialized data segment */
    char* p;                    /* Allocated in frame for main() */


    doCalc(key);

    exit(EXIT_SUCCESS);
}
```