line 5: char globBuf[65536];
Q1: Where is allocated?
A: uninitialized data

line 6: int primes[] = { 2, 3, 5, 7 };
Q2: Where is allocated?
A: initialized data

line 9: square(int x)
Q3. Where is allocated?
A: allocated in frame for square()

line 11: int result;
Q4: Where is allocated?
A: allocated in frame for square()

line 14: return result;
Q5: How the return value is passed?
A: passed via register

line 18: doCalc(int val)
Q6. Where is allocated?
A: allocated in frame for doCalc()

line 23: int t;
Q7: Where is allocated?
A: allocated in frame for doCalc()

line 31: main(int argc, char* argv[])
Q8: Where is allocated?
A: allocated in frame for main()

line 33: static int key = 9973;
Q9: Where is allocated?
A: initialized data

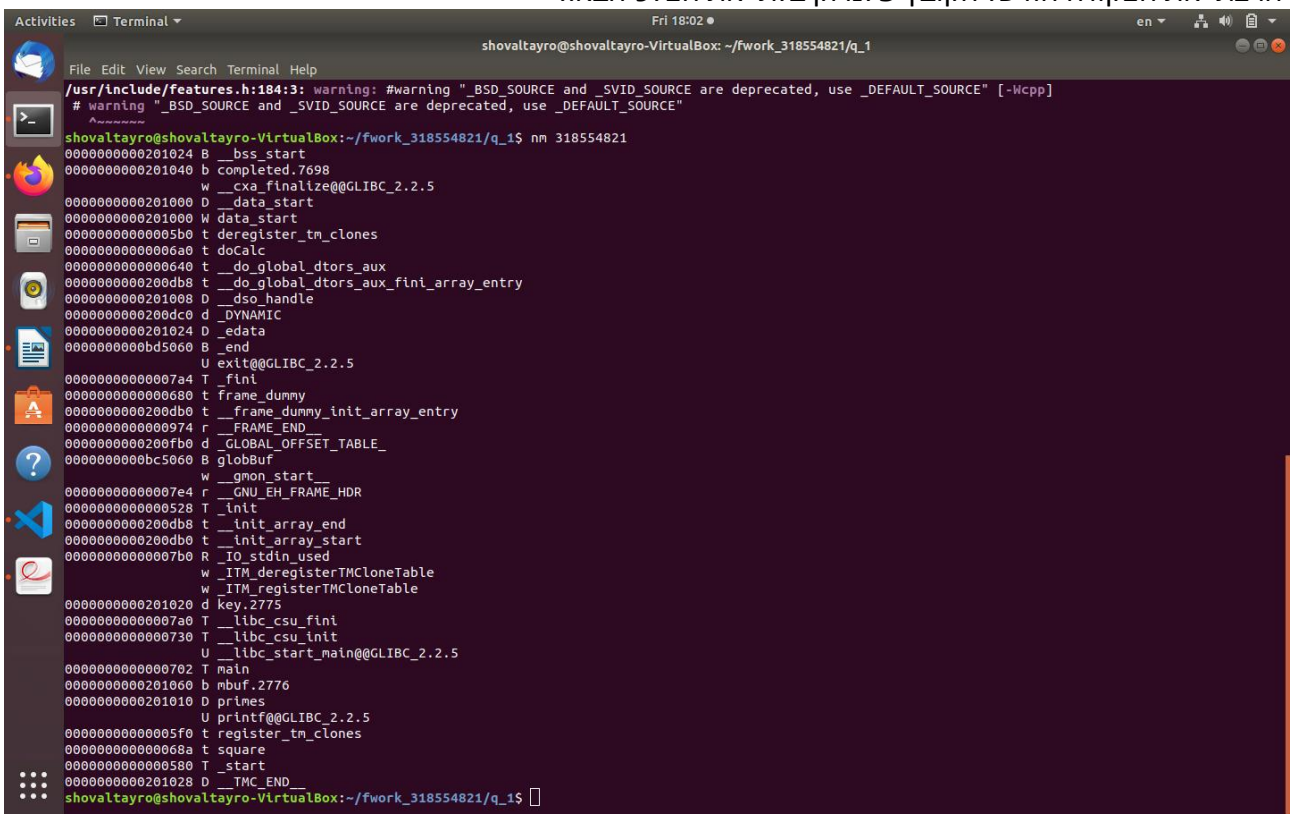
line 34: static char mbuf[10240000];
Q10: Where is allocated?
A: uninitialized data

line 35: char* p;
Q11: Where is allocated?
A: allocated in frame for main()

הפקודה nm נותנת רשימה של הסמלים המופיעים בobject file. עבור כל סמל nm מראה:

1. את ערך הסמל ברדיקס שנבחר. (הקסדצימאלי כברירת מחדל).
2. סוג הסמל, אם סוג הסמל מופיע באות קטנה – הסמל מקומי.
   אם מופיע באות גדולה – הסמל גלובלי.

הרצתי את הפקודה הזו על הקובץ שלנו וקיבלתי את הפלט הבא:.



- בסמלים **main**, **doCalc**, **square** מופיעים האותיות **'T'** או **'t'** המציינות שהסמלים האלה נמצאים ב **text section**, כלומר נוצר frame חדש עבורם.
  תשובה לשאלות 3, 6 ו-8.

- בסמלים **globBuf** ו-**mbuf** מופיעות האות **'B'** ו**'b'** בהתאמה, המציינות שהסמל הזה נמצא ב **uninitialized data.**
  תשובה לשאלות 1 ו- 10.

- בסמלים **primes** ו-**key** מופיעות האותיות **'D'** ו **'d'** בהתאמה, המציינות שהסמל נמצא ב **initialized data.**
  תשובה לשאלות 2 ו-9.

הפקודה **objdump -d** מציג מידע על object file. האפשרות של d- מציגה את פקודות המכונה, אפשרות זו מפרקת רק את החלקים שצפויים להכיל הוראות.
לאחר הרצת הפקודה קיבלתי את הפלט הבא:

```
Activities  Terminal                    Fri 19:33                    en
shovaltayro@shovaltayro-VirtualBox: ~/fwork_318554821/q_1
File  Edit  View  Search  Terminal  Help
 6a0:   55                      push   %rbp
 6a1:   48 89 e5                mov    %rsp,%rbp
 6a4:   48 83 ec 20             sub    $0x20,%rsp
 6a8:   89 7d ec                mov    %edi,-0x14(%rbp)
 6ab:   8b 45 ec                mov    -0x14(%rbp),%eax
 6ae:   89 c7                   mov    %eax,%edi
 6b0:   e8 d5 ff ff ff          callq  68a <square>
 6b5:   89 c2                   mov    %eax,%edx
 6b7:   8b 45 ec                mov    -0x14(%rbp),%eax
 6ba:   89 c6                   mov    %eax,%esi
 6bc:   48 8d 3d f1 00 00 00    lea    0xf1(%rip),%rdi        # 7b4 <_IO_stdin_used+0x4>
 6c3:   b8 00 00 00 00          mov    $0x0,%eax
 6c8:   e8 83 fe ff ff          callq  550 <printf@plt>
 6cd:   81 7d ec e7 03 00 00    cmpl   $0x3e7,-0x14(%rbp)
 6d4:   7f 29                   jg     6ff <doCalc+0x5f>
 6d6:   8b 45 ec                mov    -0x14(%rbp),%eax
 6d9:   0f af 45 ec             imul   -0x14(%rbp),%eax
 6dd:   8b 55 ec                mov    -0x14(%rbp),%edx
 6e0:   0f af c2                imul   %edx,%eax
 6e3:   89 45 fc                mov    %eax,-0x4(%rbp)
 6e6:   8b 55 fc                mov    -0x4(%rbp),%edx
 6e9:   8b 45 ec                mov    -0x14(%rbp),%eax
 6ec:   89 c6                   mov    %eax,%esi
 6ee:   48 8d 3d d7 00 00 00    lea    0xd7(%rip),%rdi        # 7cc <_IO_stdin_used+0x1c>
 6f5:   b8 00 00 00 00          mov    $0x0,%eax
 6fa:   e8 51 fe ff ff          callq  550 <printf@plt>
 6ff:   90                      nop
 700:   c9                      leaveq
 701:   c3                      retq

0000000000000702 <main>:
 702:   55                      push   %rbp
 703:   48 89 e5                mov    %rsp,%rbp
 706:   48 83 ec 10             sub    $0x10,%rsp
 70a:   89 7d fc                mov    %edi,-0x4(%rbp)
 70d:   48 89 75 f0             mov    %rsi,-0x10(%rbp)
 711:   8b 05 09 09 20 00       mov    0x200909(%rip),%eax        # 201020 <key.2775>
 717:   89 c7                   mov    %eax,%edi
 719:   e8 82 ff ff ff          callq  6a0 <doCalc>
 71e:   bf 00 00 00 00          mov    $0x0,%edi
 723:   e8 38 fe ff ff          callq  560 <exit@plt>
 728:   0f 1f 84 00 00 00 00    nopl   0x0(%rax,%rax,1)
 72f:   00

0000000000000730 <__libc_csu_init>:
```

```
Activities  Terminal                    Fri 19:33                    en
shovaltayro@shovaltayro-VirtualBox: ~/fwork_318554821/q_1
File  Edit  View  Search  Terminal  Help
000000000000068a <square>:
 68a:   55                      push   %rbp
 68b:   48 89 e5                mov    %rsp,%rbp
 68e:   89 7d ec                mov    %edi,-0x14(%rbp)
 691:   8b 45 ec                mov    -0x14(%rbp),%eax
 694:   0f af 45 ec             imul   -0x14(%rbp),%eax
 698:   89 45 fc                mov    %eax,-0x4(%rbp)
 69b:   8b 45 fc                mov    -0x4(%rbp),%eax
 69e:   5d                      pop    %rbp
 69f:   c3                      retq

00000000000006a0 <doCalc>:
 6a0:   55                      push   %rbp
 6a1:   48 89 e5                mov    %rsp,%rbp
 6a4:   48 83 ec 20             sub    $0x20,%rsp
 6a8:   89 7d ec                mov    %edi,-0x14(%rbp)
 6ab:   8b 45 ec                mov    -0x14(%rbp),%eax
 6ae:   89 c7                   mov    %eax,%edi
 6b0:   e8 d5 ff ff ff          callq  68a <square>
 6b5:   89 c2                   mov    %eax,%edx
 6b7:   8b 45 ec                mov    -0x14(%rbp),%eax
 6ba:   89 c6                   mov    %eax,%esi
 6bc:   48 8d 3d f1 00 00 00    lea    0xf1(%rip),%rdi        # 7b4 <_IO_stdin_used+0x4>
 6c3:   b8 00 00 00 00          mov    $0x0,%eax
 6c8:   e8 83 fe ff ff          callq  550 <printf@plt>
 6cd:   81 7d ec e7 03 00 00    cmpl   $0x3e7,-0x14(%rbp)
 6d4:   7f 29                   jg     6ff <doCalc+0x5f>
 6d6:   8b 45 ec                mov    -0x14(%rbp),%eax
 6d9:   0f af 45 ec             imul   -0x14(%rbp),%eax
 6dd:   8b 55 ec                mov    -0x14(%rbp),%edx
 6e0:   0f af c2                imul   %edx,%eax
 6e3:   89 45 fc                mov    %eax,-0x4(%rbp)
 6e6:   8b 55 fc                mov    -0x4(%rbp),%edx
 6e9:   8b 45 ec                mov    -0x14(%rbp),%eax
 6ec:   89 c6                   mov    %eax,%esi
 6ee:   48 8d 3d d7 00 00 00    lea    0xd7(%rip),%rdi        # 7cc <_IO_stdin_used+0x1c>
 6f5:   b8 00 00 00 00          mov    $0x0,%eax
 6fa:   e8 51 fe ff ff          callq  550 <printf@plt>
 6ff:   90                      nop
 700:   c9                      leaveq
 701:   c3                      retq

0000000000000702 <main>:
 702:   55                      push   %rbp
 703:   48 89 e5                mov    %rsp,%rbp
```

- ניתן לראות שבתוך הframe של square אנו דוחפים את result לתוך רגיסטר rbp ובסוף הפונקציה אנחנו עושים לו pop ומחזירים אותו. לכן הסקתי כי **result מועבר ע"י רגיסטר.**
תשובה לשאלה 5.

- אנו יודעים כי לכל frame יש מחסנית זיכרון משלו שמוקצה לו בקריאה לפונקציה ומשוחררת כאשר הפונקציה (הframe) נגמרת. הפקודה objdump מראה את פקודות המכונה לפי ה frame שלהם, כלומר כל frame והפקודות הקורות בו. ניתן לראות כי בכל אחת מהפונקציה doCalc, square ו-main כאשר מגדירים משתנים בתוך הפונקציה מבוצעת הפקודה Push. הפקודה push מלמדת על כך שבוצע דחיפה של משתנה למחסנית של הפונקציה, ערך זה ימחק ברגע שהפונקציה תסתיים מאחר והוגדר על המחסנית אשר משוחררת בסוף הפונקציה (בסוף הframe).
תשובה לשאלות 4, 7 ו-11.