

## הסבר מפורט על התשובות:

### הסבר מקדים על שימוש בפקודות :

הסבר על bss section - The **BSS segment** contains all global variables and static variables that are initialized to zero or do not have explicit initialization in source code

שימוש בפקודת **nm** – הפקודה מביאה לנו symbols בקובץ מסוג objfile. ה symbols הרלוונטים שמצאתי לנכון כשימושיים במטלה עצמה הם :

"B"/"b" – symbol שמראה שמשנתה מוקצה על ה bss section כלומר על איזור קוד של משתנים שלא אותחלו. תמונת מקור:

```
"B"
"b" The symbol is in the BSS data section. This section typically contains zero-initialized or uninitialized data, although the exact behavior is system dependent.
```

"D"/"d" – symbol שמראה שמשנתה מוקצה על ה data section כלומר על איזור קוד של משתנים שאותחלו. תמונת מקור:

```
"D"
"d" The symbol is in the initialized data section.
```

"T"/"t" – symbol שמראה שההקצאה מתבצעת על ה text section. תמונת מקור:

```
"T"
"t" The symbol is in the text (code) section.
```

שימוש בפקודת **objdump** – הפקודה מראה לנו את קוד האסמבלי של קובץ הרצה מסוים, יש כמה אופציות:

1. לראות את קוד האסמבלי של text section
2. לראות את קוד האסמבלי של data section
3. לראות את קוד האסמבלי של bss section

דוגמא:

```
/Desktop/eliel/OSProjects/fwork_316519966/q1$ objdump -j .text -D objectfile
```

שימוש בפקודת **size** – הפקודה מראה לנו לפי sections את כמות הבתים שיש בקובץ object לפי המשתנים והפונקציות שהוקצו. דוגמא:

```
eliel@eliel-mint:~/Desktop/eliel/OSProjects/fwork_316519966/q1$ size objectfile
text      data      bss       dec        hex filename
1829      628 10305568   10308025   9d49b9 objectfile
```

### תשובה מפורטת לשאלה 1:

לאחר הרצה של הפקודת nm על הקובץ objectfile שלי, גיליתי כי מופיע ה B symbol ליד המשתנה שנקרא globBuf, כלומר משתנה זה נמצא באיזור bss, משתנה שלא אותחל.

תמונת מקור: (nm)

```
0000000000200d00 t __frame_dummy_init_array_e
0000000000000974 r __FRAME_END__
0000000000200fb0 d __GLOBAL_OFFSET_TABLE__
0000000000bc5060 B globBuf
w __gmon_start__
0000000000000000 t __gnu_eh_frame_hdr
```

### תשובה מפורטת לשאלה 2:

לאחר הרצה של הפקודת nm על הקובץ objectfile שלי, גיליתי כי מופיע ה D symbol ליד המשתנה שנקרא primes, כלומר משתנה זה נמצא באיזור data, משתנה שאותחל.

תמונת מקור: (nm)

```
0000000000201010 D primes
```

### תשובה מפורטת לשאלה 3:

לאחר הרצה של הפקודת nm על הקובץ objectfile שלי, גיליתי כי מופיע ה t symbol ליד הפונקציה שנקראת square, כלומר ההקצאה התבצעה על איזור ה text

תמונת מקור: (nm)

```
000000000000068a t square
```

### תשובה מפורטת לשאלה 4:

לאחר הרצה של הפקודת objdump באיזור ה text כי שם נמצא הפונקציה square לפי תשובה 3, נמצא כי כל השימוש במשתנה result נמצא אך ורק בתוך ה scope של square. ואין הקצאה דינמאית לכן זה יוקצה על המחסנית (stack)

תמונת מקור: (objdump)

```
000000000000068a <square>:
68a: 55                push    rbp
68b: 48 89 e5          mov     rbp,rsi
68e: 89 7d ec          mov     DWORD PTR [rbp-0x14],edi
691: 8b 45 ec          mov     eax,DWORD PTR [rbp-0x14]
694: 0f af 45 ec       imul    eax,DWORD PTR [rbp-0x14]
698: 89 45 fc          mov     DWORD PTR [rbp-0x4],eax
69b: 8b 45 fc          mov     eax,DWORD PTR [rbp-0x4]
69e: 5d               pop     rbp
69f: c3               ret
```

נשים לב שהכתובות הוירטואליות זהות גם בהרצה של nm וגם ב objdump (000000000000068a) פירוש קוד האסמבלי:

3 שורות ראשונות מבצעות הקצאת זיכרון לפרמטר x שמקבלת הפונקציה.  
3 שורות הבאות מתבצעות הקצאה למשתנה result והשמה של הערך x\*x לתוך המשתנה.  
שורה אחת אחרי מעביר את הערך לתוך רגיסטר eax

ואז מתבצע ret שמחזיר ערך מהפונקציה.  
ההסבר הזה יעזור לי לתשובה לשאלה 5.

### תשובה מפורטת לשאלה 5:

לאחר הרצה של הפקודות objdump על ה objectfile שלי מצאנו את קוד האסמבלי של square (תשובה לשאלה 4) ובנוסף נמצא גם את קוד האסמבלי של הקריאה לאותה פונקציה square שנמצאת בתוך doCalc. ונראה שבנוסף להסבר בתשובה לשאלה 4, הערך המוחזר מהפונקציה square מועבר ע"י רגיסטר eax.

### תמונת מקור: (objdump)

```
6ab: 8b 45 ec      mov     eax,DWORD PTR [rbp-0x14]
6ae: 89 c7         mov     edi,eax
6b0: e8 d5 ff ff   call    68a <square>
6b5: 89 c2         mov     edx,eax
6b7: 8b 45 ec      mov     eax,DWORD PTR [rbp-0x14]
```

### תשובה מפורטת לשאלה 6:

לאחר הרצה של הפקודות nm על הקובץ objectfile שלי, גיליתי כי מופיע ה symbol t ליד הפונקציה שנקראת doCalc, כלומר ההקצאה התבצעה על איזור ה text

### תמונת מקור: (nm)

```
0000000000000000 t _deregister
000000000000006a t doCalc
0000000000000000 t _deregister
```

### תשובה מפורטת לשאלה 7:

לאחר הרצה של הפקודות objdump על ה objectfile שלי באיזור ה text כי שם נמצא הפונקציה square לפי תשובה 6, נמצא כי כל השימוש במשתנה t נמצא אך ורק בתוך ה scope של doCalc. ואין הקצאה דינאמית לכן זה יוקצה על המחסנית (stack) לפי התמונת מקור כל ההתעסקות עם המשתנה t נמצא אך ורק בתוך ה scope של doCalc, שם אנחנו מכפילים 3 פעמים במשתנה val (לפי האסמבלי יש פעמיים imul, כי מכפילים val \* val ואז את התוצאה בעוד val אחד)

### תמונת מקור: (objdump)

```
6d6: 8b 45 ec      mov     eax,DWORD PTR [rbp-0x14]
6d9: 0f af 45 ec   imul    eax,DWORD PTR [rbp-0x14]
6dd: 8b 55 ec      mov     edx,DWORD PTR [rbp-0x14]
6e0: 0f af c2      imul    eax,edx
6e3: 89 45 fc      mov     DWORD PTR [rbp-0x4],eax
6e6: 8b 55 fc      mov     edx,DWORD PTR [rbp-0x4]
```

### תשובה מפורטת לשאלה 8:

לאחר הרצה של הפקודות nm על הקובץ objectfile שלי, גיליתי כי מופיע ה T symbol ליד הפונקציה שנקראת main, כלומר ההקצאה התבצעה על איזור ה text

### תמונת מקור: (nm)

```
0000000000000702 T main
```

### תשובה מפורטת לשאלה 9:

לאחר הרצה של הפקודת nm על הקובץ objectfile שלי, גיליתי כי מופיע ה symbol d ליד המשתנה שנקרא key, כלומר משתנה זה נמצא באיזור data, משתנה שאותחל.

תמונת מקור: (nm)

```
w _ITM_registerTMCloneTable
000000000000201020 d key.2775
000000000000007a0 T _libc_csu_fini
```

### תשובה מפורטת לשאלה 10:

לאחר הרצה של הפקודת nm על הקובץ objectfile שלי, גיליתי כי מופיע ה symbol b ליד המשתנה שנקרא mbuf, כלומר משתנה זה נמצא באיזור bss, משתנה שלא אותחל.

תמונת מקור: (nm)

```
00000000000000702 T main
000000000000201060 b mbuf.2776
```

### תשובה מפורטת לשאלה 11:

לאחר הרצה של הפקודת objdump על ה objectfile שלי באיזור ה text כי שם נמצא הפונקציה main לפי תשובה 8, נמצא כי כל השימוש במשתנה p נמצא אך ורק בתוך ה scope של main. לפי התמונת מקור מתבצעת הקצאה דינאמית לתוך המשתנה p.

תמונת מקור: (objdump)

```
711: 8b 05 09 09 20 00    mov     eax,DWORD PTR [rip+0x200909]
717: 89 c7               mov     edi,eax
```

### תשובה מפורטת לשאלה 12:

לאחר הרצה של הפקודת objdump על ה objectfile שלי באיזור ה text כי שם נמצא הפונקציה main לפי תשובה 8, כי ההקצאה לתוך המשתנה p היא הקצאה דינאמית ע"י קריאה לפונקציה malloc.

לפי התמונת מקור נוכל לראות את ההקצאה הזאת ע"י שימוש במילה rip, שזה בעצם relative address.

```
711: 8b 05 09 09 20 00    mov     eax,DWORD PTR [rip+0x200909]
717: 89 c7               mov     edi,eax
```