

## הסביר מפורט על התשובות:

### **הסביר מקדימ שימוש בפקודות :**

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

שימוש בפקודה **nm** – הפקודה מביאה לנו **symbols** בקובץ מסוג **objfile** . ה **symbols** הרלוונטיים שמצאתι לנו כשימושים במתלה עצמה הם :

"b"/"B" – שマーה משתנה מוקצתה על ה **bss section** יכולת על איזור קוד של משתנים שלא אותו כלום. **תמונה מקורה:**

```
"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" – שマーה משתנה מוקצתה על ה **data section** יכולת על איזור קוד של משתנים שאותוכלו. **תמונה מקורה:**

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

.text section – שマーה שההקצתה מתבצעת על ה **.text section** **תמונה מקורה:**

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

שימוש בפקודה **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 בפקודה **size** – הפקודה מראה לנו לפי sections את כמות הבטים 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 לשאלת 1:

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

תמונה מקור: (nm)

```

a_gmon_start          w
GLOBAL_OFFSET_TABLE_  d
FRAME_END             r
globBuf               B

```

### תשובה מפורטת לשאלת 2:

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

תמונה מקור: (nm)

```

primes                D

```

### תשובה מפורטת לשאלת 3:

לאחר הריצה של הפקודת `nm` על הקובץ objectfile שלי, גיליתי כי מופיע ה symbol t ליד הפונקציה שנקראת `square`, כלומר ההפונקציה הטעינה על איזור ה `text`.

תמונה מקור: (nm)

```

square                t

```

### תשובה מפורטת לשאלת 4:

לאחר הריצה של הפקודת `objdump -t` בAIOR ה `text` Ci שם נמצא הפונקציה `square` לפני תשובה 3, נמצא Ci כל השימוש במשתנה `result` נמצא אך ורק בתחום ה `scope` של `square`. אין ההפונקציה דינמית لكن זה יוקצה על המחסנית (`stack`)

תמונה מקור: (objdump)

```

0000000000000068a <square>:
    68a: 55                      push   rbp
    68b: 48 89 e5                mov    rbp, rsp
    68c: 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 הווירטואליות זהות גם בהרצה של `nm` וגם ב `objdump` (0000000000000068a).  
פירוש קוד האסמלבי:

3 שורות ראשונות מבצעות הקצת זיכרון לפרמטר x שמקבלת הפונקציה.  
3 שורות הבאות מטבח ההפונקציה `result` למשתנה result והשמה של הערך x\*ax לתוך המשתנה.  
שורה אחת אחריה מעביר את הערך לתוך גירסת x eax

ואז מתבצע ret שמחזיר ערך מהפונקציה.  
הסביר זה יעזר לך לתשובה לשאלה 5.

#### תשובה מפורטת לשאלה 5:

לאחר הריצה של הפקודת objdump על ה objectfile של' מצאנו את קוד האסמלבי של square (תשובה לשאלה 4) ובנוסף נמצא גם את קוד האסמלבי של הקראיה לאותה פונקציה square שנמצאת בתוך doCalc . ונראה שבנוסף להסביר בתשובה לשאלה 4 , הערך המוחזר מהפונקציה square מועבר ע"י רגיסטר eax .

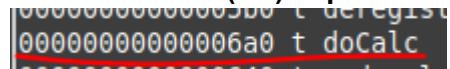
#### תמונה מקור: (objdump)

```
6ab: 8b 45 ec          mov    eax,DWORD PTR [rbp-0x14]
6ae: 89 c7              mov    edi, eax
6b0: e8 d5 ff 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 הריצה של הפקודת nm על הקובץ objectfile של' , גיליתי כי מופיע ה symbol t ליד הפונקציה שנקרהת doCalc , ככלומר ההקראה התבכעה על איזור ה text

#### תמונה מקור: (nm)



000000000000006a0 t doCalc

#### תשובה מפורטת לשאלה 7:

לאחר הריצה של הפקודת objdump על ה objectfile של' בAIZOR ה text Ci שם נמצא הפונקציה square לפי תשובה 6, נמצא Ci כל השימוש במשתנה t נמצא אך ורק בתחום ה scope של doCalc . ואין הקראה דינמית לכך זה יוקצה על המחסנית ( stack )

לפי התמונה מקור כל ההתעקחות עם המשתנה t נמצא אך ורק בתחום ה scope של doCalc , שם אנחנו מכפילים 3 פעמים במשתנה val (לפי האסמלבי יש פעמיים imul Ci מכפילים val \* val ואז את התוצאה בעוד val אחד)

#### תמונה 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 הריצה של הפקודת nm על הקובץ objectfile של' , גיליתי כי מופיע ה symbol T ליד הפונקציה main שנקרהת main , ככלומר ההקראה התבכעה על איזור ה text

#### תמונה מקור: (nm)



00000000000000702 T main

### תשובה מפורטת לשאלה 9:

לאחר הריצה של הפקודת `nm` על הקובץ objectfile שלו , גיליתי כי מופיע ה symbol `d` ליד המשתנה שנקרא `key` , כלומר משתנה זה נמצא באיזור `data` , משתנה שלא אומתחל.

תמונה מקור: (nm)

```
w ITM registerTMCloneTable
000000000000201020 d key.2775
0000000000000000/a0 T libc csu fini
```

### תשובה מפורטת לשאלה 10:

לאחר הריצה של הפקודת `nm` על הקובץ objectfile שלו , גיליתי כי מופיע ה symbol `b` ליד המשתנה שנקרא `mbuf` , כלומר משתנה זה נמצא באיזור `bss` , משתנה שלא אומתחל.

תמונה מקור: (nm)

```
0000000000000000/02 T main
000000000000201060 b mbuf.2776
0000000000000000/00 T _init
```

### תשובה מפורטת לשאלה 11:

לאחר הריצה של הפקודת `objdump` על ה objectfile שלו בAIIZOR ה `text` כי שם נמצא הפונקציה `main` לפני תשובה 8 , נמצא כי כל השימוש במשתנה `k` נמצא אך ורק בתחום ה `scope` של `main` .  
לפי התמונה מקור מתבצעת הקזאה דינמית לתוך המשתנה `k` .

תמונה 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 הריצה של הפקודת `objdump` על ה objectfile שלו בAIIZOR ה `text` כי שם נמצא הפונקציה `main` לפני תשובה 8 , כי ההקזאה לתוך המשתנה `k` היא הקזאה דינמית ע"י קרייה לפונקציה `malloc` .  
לפי התמונה מקור נוכל לראות את ההקזאה 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
```