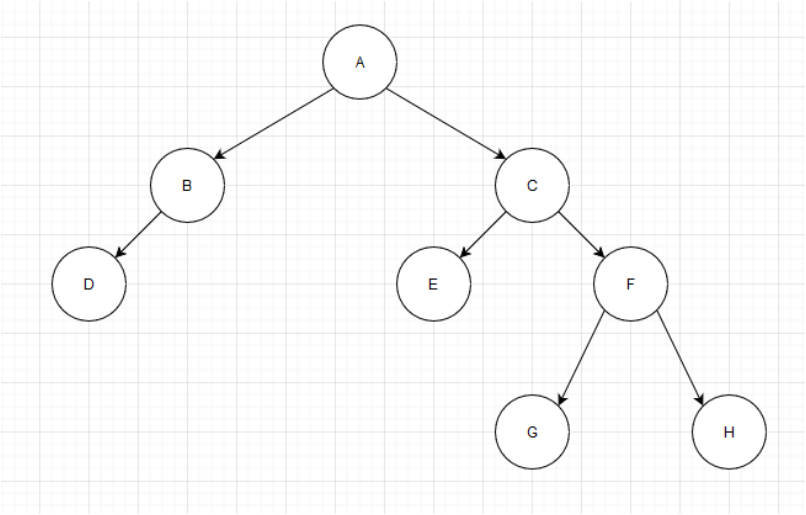
את"מ – ת"ב 2

שאלה 1:

א. הגרף הנוצר הינו:



ב. כן – הקוד עומד בקוננציות של System V.

ג. נניח שבתחילת התוכנית ערך RSP הוא X. לכן, הערך המקסימלי שאליו יגיע RSP במהלך ריצת התוכנית הוא:

$$X - 8 \cdot 5 = X - 40$$

ד. הפלט יהיה true, כלומר 1 = %eax.

ה. נשלים את הקוד:

```
int func (Node *root, int x){
    if (root->data == x )
        return 1;
    if (root->left != null)
        If ( func(root->left, x) == 1 )
            return 1;
    if (root->right != null)
        return func(root-> right, x);
    return 0;
}
```

ו. תחילה, הקוד המקורי:

```
.section .text
.global _start
_start:
    mov $8, %esi
    mov $A, %rdi
    call func
    movq $60, %rax
    movq $0, %rdi
    syscall

func:
    pushq %rbp
    movq %rsp, %rbp
    cmp (%rdi), %esi
    jne continue
    mov $1, %eax
    jmp finish

continue:
    cmpq $0, 4(%rdi)
    je next
    pushq %rdi
    mov 4(%rdi), %rdi
    call func
    pop %rdi
    cmp $1, %eax
    je finish
next:
    cmpq $0, 12(%rdi)
    je fail
    pushq %rdi
    mov 12(%rdi), %rdi
    call func
    pop %rdi
    cmp $1, %eax
    je finish

fail:
    mov $0, %eax
finish:
    leave
    ret
```

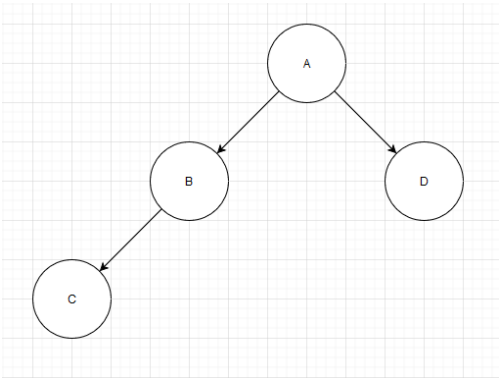
השינויים:

- שורה 30 – mov $8, **%rsi**
- שורה 40 – cmp (%rdi), **%rsi**
- שורה 46 – cmpq $0, **8**(%rdi)
- שורה 49 – mov **8**(%rdi), %rdi
- שורה 55 – cmpq $0, **16**(%rdi)
- שורה 58 – mov **16**(%rdi), %rdi

ז. התשובה היא D – התוכנית תקרוס במהלך הריצה.
למעשה הקוד הוא חיפוש preorder בעץ עבור איבר שמכיל את המספר 8. אם נמצא איבר כזה הפונקציה מחזירה 1, אחרת 0. כפי שניתן לראות מהאיור בסעיף א', הצומת D נמצא בקצה השמאלי של העץ הנוצר על פי קוד האסמבלי (בהנחה כי המצביע הראשון לצומת אחר מצומת כלשהו הוא אכן מצביע לבן השמאלי, כמו בסעיף ה').
אם נצביע לצומת A כבן שמאלי של הצומת D, אז נקבל כי אכן תתקבל לנו לולאה אינסופית ונמשיך לרוץ עד שהתוכנית תקרוס. התוכנית תקרוס מכיוון שבמהלך הריצה על התוכנית אנו עושים push למחסנית בכל קריאה לפונקציה מחדש, אך איננו מגיעים ל – pop בשום מצב בעקבות הלולאה האינסופית.
לכן בשלב כלשהו ייגמר לנו המקום בזכרון והתוכנית תקרוס.

ח. נסביר:

1. **אין פגיעה בנכונות.** כיוון שהפעולה נעשית על עץ בינארי, אין חשיבות לשמירתו ושחזורו של RDI כאשר מגיעים ל next שכן משם החזרה בסוף הסיור תהיה אל השורש/אל סוף התוכנית.
2. **אין פגיעה בנכונות**. אמנם נגיע למצב שבו אנו מכניסים איברים רבים למחסנית ולא נראה שנשחרר אותם, אך למעשה RSP יתעדכן בסוף סיור על כל תת עץ ולכן יצביע על האיבר הרצוי בסוף הפעולה, והתוכנית תמשיך כצפוי.
3. **ישנה פגיעה בנכונות**. נתבונן בקלט הבא:
נחפש את הצומת D.
נשים לב שאם לא נשמור את רגיסטר RDI בשורות אלו כשנפנה שמאלה בעץ, נאבד את השורש A ולכן לא נוכל לפנות ימינה בחיפוש לאחר מכן, ונקבל פלט שגוי.



4. **אין פגיעה בנכונות.** אמנם ננצל יותר משטח המחסנית, אך בדומה לסעיף השני, לאחר סיום הסיור בתת עץ, במהלך פקודת LEAVE נשחזר את RSP להצביע לערך הנכון.
5. **אין פגיעה בנכונות.** כבר ראינו ששני השינויים האחרונים המצויינים לא פוגעים בנכונות (סעיף ראשון), ובנוסף בסעיף הקודם ראינו שהשינוי הראשון המצויין לא פוגע בנכונות. לכן נתבונן בשינוי: שינוי פקודת הPOP שבשורה 51.
למעשה שינוי זה ישחזר את ערך RDI כרצוי, אך לא יעדכן את RSP בהתאם ולכן יהיה ערך זבל במחסנית. בדומה לסעיף הקודם, זה לא מפגיע לנו כי RSP ישוחזר כרצוי לאחר LEAVE.

שאלה 2:

א. נגדיר כך:

msg1_len: .quad msg2 – msg1
msg2_len: .quad msg1_len – msg2
all_msg_len: .quad msg1_len – msg1

ב. בסיום ריצת הקוד יודפס: HWYOUDON.
ג. כעת יודפס: hwyoudonje@osG@hr@od.
ד. כעת יודפס: hwyoudonje@osG@hr@od0.
ה. נפתור:
1. הקוד יעבוד כי אין הבדל משמעותי בין הרגיסטרים מאחר ו - syscall לא משתמשת בהם.
2. הקוד לא יעבוד מאחר ו - r11 נדרס ע"י RFLAGS כאשר syscall נקרא.
3. הקוד לא יעבוד מאחר ו - rdi משמש כ - descriptor של הפלט ולכן אסור לשנות אותו מהערך 1 אחרת לא יודפס למסך כלום.
4. הקוד לא יעבוד כיוון ש - rcx נדרס ע"י rip.
5. הקוד יעבוד כיון שאין שימוש במחסנית בקוד הנ"ל אך זה לא מומלץ כמובן ולא עומד בקונבציות.

ו. יודפס h ולאחר מכן נכנס ללואה אינסופית, כיוון שכמו שהזכרנו בסעיף הקודם, rcx נדרס ע"י rip ולכן ברגע שנקרא ל - syscall ה - rip שכרגע מצביע על skip ייכנס לתוך rcx ובכל פעם שנעשה jmp *rcx נכנס ל - skip ללא אפשרות ליציאה מהלולאה.

שאלה 3:

א. rax = 1:

$$x = 0x7F$$

rax = 2:

$$0xFF \geq x \text{ או } x \geq 0x80$$

rax = 3:

$$0x0 \leq x \leq 0x7E$$

ב. תחילה נעשה pushfq. כעת נרצה להשתמש בפקודת OR על מקום זה בזכרון עם הערך 0x801. ערך זה מסמן ששני הדגלים OF ו-CF דלוקים יחד. לאחר מכן תעשה popfq וסיימנו. בפסודו קוד:

```
pushfq
orq $0x801, (%rsp)
popfq
```

ג. כדי לעשות זאת היא תצטרך לתת לעצמה גישה בתור משתמש לרמת הקרנל, כלומר היא תרצה לשנות את הביטים 12 ו-13 ב - RFLAGS שמציינים I/O Privilege Level ל – 11 וכך תביא לעצמה גישה כמו שרצתה.
ד. על מנת להשתמש בפקודה popfq נדרשת רמת הרשאה של הגרעין, כלומר CPL=0. אין באפשרותה לקבל רמת הרשאה זו ללא שימוש בפקודה SYSCALL שמעבירה את ההרשאה מרמת משתמש לרמת גרעין. מעבר לכך, קוד הגרעין נכתב בצורה כזו למניעת פרצות אבטחה חמורות, ולכן ברגע שחוזרים

מקוד הגרעין לקוד המשתמש מעדכנים את CPL להיות שוב 3, ובכך נשללת הגישה של המשתמש אל דברים בהם לא אמור לגעת.