תרגיל 7 – Bitwise, Files

תאריך הגשה: 24.1.18 משקל התרגיל: 5%

חלק א' – Bitwise

נתונות לכם הצהרות של פונקציות בצורה הבאה (בתוך קובץ funcs.h – מצורף לתרגיל):

```
int function(arg1, arg2, ...);
```

עליכם לממש את הפונקציות באופן הבא (בתוך קובץ בשם funcs.c):

```
int function(arg1, arg2, ...) {
   int VarA = ExprA;
   ...
   VarJ = ExprJ;
   ...
   VarN = ExprN;
   return ExprR;
}
```

כאשר:
בכל מקום שמופיע בו Var הכוונה היא למשתנה מקומי של הפונקציה.
בכל מקום שמופיע בו Expr הכוונה היא לאחד מהדברים הבאים, או שילוב שלהם:

1. קבוע שלם מ0 עד 255 (0xFF). אין להשתמש בקבועים גדולים יותר. כאשר אתם משתמשים בקבוע כמסכת ביטים (bitmask) כתבו אותו בבסיס 16, אחרת כתבו אותו בבסיס 10 כרגיל.
הקבועים יכולים להופיע בתוך הקוד, אין צורך להגדיר אותם בעזרת const\define בתרגיל הזה.
2. פרמטר של הפונקציה (למשל arg1).
3. משתנה מקומי שהגדרתם בתוך הפונקציה.
4. פעולת NOT לוגי (!) ופעולת חיבור (+).
5. פעולות Bitwise (<< >> ^ | & ~).
6. סוגריים עגולים.

- אין להשתמש בפעולות נוספות (למשל && || * / - ? == וכו').
- כמובן שמותר להשתמש בפעולת ההשמה (=).
- בנוסף, אין להשתמש בתנאים (if, switch), לולאות (for, do, while), להגדיר מקרו (define), להשתמש בפונקציות עזר (כולל פונקציות מתוך התרגיל), לבצע casting ולהשתמש בסוגים נוספים של משתנים שאינם int.

הניחו כי התרגיל ייבדק בסביבת עבודה שמשתמשת בשיטת 2's complement עם 32 ביטים לייצוג מספרים שלמים (השרת u2).

תזכורת: כדי להדפיס בעזרת printf את המספרים בבסיס 16 יש לכתוב %x או %X

```
int var = 51;
printf("%x\n", var); // output is: 33 (which is: 0011 0011 = 1+2+16+32 = 51)
```

```
int var = 0x3E;       // which is 0011 1110 = 2+4+8+16+32 = 62
printf("%d\n", var); // output is: 62
```

עליכם לממש את הפונקציות הבאות תוך התחשבות גם בהגבלות הנתונות עבור כל פונקציה.
עליכם לתעד כל שורה בקוד ולהסביר מדוע אתם מבצעים כל פעולה. התיעוד יתבצע בעזרת הערות שורה (//) ויהיה באנגלית בלבד (אם עבור שורה מסויימת ההסבר שלכם מורכב, ניתן להסביר בעזרת הערת בלוק שתופיע לפניה. אם ההסבר מאוד מורכב, ניתן לפצל את השורה לכמה שורות פשוטות יותר). אם אתם מייצרים bitmask, תיעוד מספיק יהיה לכתוב את הייצוג הבינארי המלא.

שלוש הפונקציות הראשונות פתורות, אחת תואמת את הגדרות התרגיל והשתיים האחרות לא.
אין צורך לממש אותן בתרגיל שלכם, הן נמצאות כאן כדי שתבינו את הדרישות ותשימו לב להגבלות.

0. ממשו את הפונקציה pow2plus1 המקבלת משתנה x ומחזירה את 2^x + 1. ניתן להניח כי 0 <= x <= 31.
מספר פעולות מותרות: 2.
פתרון:

```
int pow2plus1(int x) {
  return (1 << x) + 1;
}
```

0. ממשו את הפונקציה swapOddEvenBits המקבלת משתנה x ומחזירה אותו כאשר כל זוג ביטים מוחלפים ביניהם (0 מוחלף עם 1, 2 מוחלף עם 3 וכו').
מספר פעולות מותרות: 5.
פתרון:
שימו לב! זהו מימוש שאינו עונה על הגדרות התרגיל, בתרגיל הזה אין אפשרות להשתמש בקבועים בגודל כזה. חשבו כיצד ניתן לבצע את אותו הדבר בעזרת הקבועים 0x55 0xaa (וכמובן, תוך שימוש במשתני עזר וביותר מ5 פעולות).

```
int swapOddEvenBits(int x) {
  return ( ((x & 0xaaaaaaaa) >> 1) | ((x & 0x55555555) << 1) );
}
```

0. ממשו את הפונקציה isPowerOf2 המקבלת משתנה x ומחזירה 1 אם הוא חזקה של 2 ומחזירה 0 אחרת.
פתרון:
שימו לב! זהו מימוש שאינו עונה על הגדרות התרגיל, בתרגיל הזה אין אפשרות להשתמש באופרטור ==

```
int isPowerOf2(int x) {
  return ( (x & (n-1)) == 0);
}
```

1. ממשו את הפונקציה int bitAnd(int x, int y); המקבלת שני משתנים x,y ומחזירה את x&y.
מותר להשתמש באופרטורים ~ | בלבד.
מספר פעולות מותרות: 8.

bitAnd(6, 5) = 4

2. ממשו את הפונקציה int bitCount(int x); המקבלת משתנה x ומחזירה את מספר הביטים שהם 1.
מספר פעולות מותרות: 40

bitCount(7) = 3 bitCount(5) = 2

3. ממשו את הפונקציה int bang(int x); המקבלת מספר x ומחשבת עבורו NOT לוגי.
מותר להשתמש באופרטורים << >> + | ^ & ~ בלבד.
מספר פעולות מותרות: 12

bang(0) = 1 bang(3) = 0

4. ממשו את הפונקציה int fitsBits(int x, int n); המקבלת שני משתנים x,n ומחזירה 1 אם ניתן לייצג את x בעזרת n ביטים, ו0 אחרת. ניתן להניח 1 <= n <= 32 וכן יש להתייחס לx כשלם חיובי (unsigned) כלומר כל 32 הביטים הם בעלי משקל חיובי.
מספר פעולות מותרות: 15

fitsBits(4,2) = 0 fitsBits(5,3) = 1

5. ממשו את הפונקציה int divpwr2(int x, int n); המקבלת שני משתנים x,n ומחזירה את x/(2^n) מעוגל לכיוון המספר 0. ניתן להניח 0 <= n <= 30
מספר פעולות מותרות: 15

divpwr2(33,4) = 2 divpwr2(15,1) = 7

6. ממשו את הפונקציה int negate(int x); המקבלת משתנה x ומחזירה את –x (המספר הנגדי).
מספר פעולות מותרות: 5

negate(1) = -1 negate(-1) = 1

7. ממשו את הפונקציה int isPositive(int x); המקבלת משתנה x ומחזירה 1 אם x > 0 ומחזירה 0 אחרת.
מספר פעולות מותרות 8

isPositive(17) = 1 isPositive(-21) = 0 isPositive(0) = 0

8. ממשו את הפונקציה int isLessOrEqual(int x, int y); המקבלת שני משתנים x,y ומחזירה 1 אם x<=y ומחזירה 0 אחרת.
מספר פעולות מותרות: 24

isLessOrEqual(4,5) = 1 isLessOrEqual(6,5) = 0 isLessOrEqual(5,5) = 1

חלק ב' – Files

כעת עליכם לכתוב את הקובץ ex7.c – תפקידו להריץ את התוכנית על פי קובץ פרמטרים, ולכתוב את התוצאות לקובץ פלט חדש.

קובץ הקלט יהיה קובץ טקסט שבו כל שורה מייצגת קריאה לאחת הפונקציות שהוגדרו בחלק הקודם. כל שורה תיפתח בשם הפונקציה שיש להפעיל, לאחר מכן התו ":", לאחר מכן מספר הפרמטרים שהפונקציה מצפה לקבל, לאחר מכן התו ":", ולאחר מכן הפרמטרים של הפונקציה, מופרדים בפסיקים.

לדוגמה:

```
bitAnd:2:6,5
bitCount:1:7
bang:1:3
fitsBits:2:8,3
divpwr2:2:33,4
negate:1:9
isPositive:1:0
isLessOrEqual:2:8,9
```

התוכנית תחלץ את הנתונים מתוך הקובץ ותייצר קובץ פלט חדש שבו עבור כל הרצה תופיע שורת פלט המורכבת משם הפונקציה, לאחר מכן התו ")", לאחר מכן הפרמטרים של הפונקציה (מופרדים בפסיקים אם יש יותר מאחד), לאחר מכן התו "(", לאחר מכן רווח, התו "=", רווח נוסף, ותוצאת ההרצה.

לדוגמה (קובץ הפלט התואם להרצת התוכנית עבור קובץ הקלט בדוגמה הקודמת):

```
bitAnd(6,5) = 4
bitCount(7) = 3
bang(3) = 0
fitsBits(8,3) = 0
divpwr2(33,4) = 2
negate(9) = -9
isPositive(0) = 0
isLessOrEqual(8,9) = 1
```

שימו לב שקובץ הפלט מכיל שורה ריקה בסופו, זה אמור להקל עליכם (זהו האנטר של הדפסת השורה הקודמת).

- התוכנית תקבל את שמות הקבצים כפרמטרים, ניתן להניח שכל הערכים יהיו תקינים (שמות הקבצים והתוכן שלהם). פקודת הרצה לדוגמה: ./a.out input.txt output.txt.

- יש לכתוב את הקוד באופן כללי ככל האפשר, כך שאם נרצה להוסיף בעתיד פונקציות לקובץ funcs.c נצטרך להוסיף רק כמה קבועים בראש הקובץ ex7.c. כדי לעשות זאת מומלץ להגדיר קבועים עבור מספר הפונקציות, עבור שמות הפונקציות ועבור מספר הפרמטרים המקסימלי שניתן לשלוח לפונקציה.

שאלה למחשבה (אין צורך להגיש משהו):

אתם לכודים בחדר עם פסיכופת רצחני. בחדר יש שולחן עגול מסתובב עם ארבעה מתגי לחיצה - לא ניתן לדעת אם מתג נמצא בOn או בOff.
באמצע השולחן יש נורה, אשר המתח עבורה מחובר למפסק ראשי נפרד. בהתחלה הנורה כבויה. ידוע שהנורה הזאת נדלקת רק כאשר כל המתגים נמצאים בOn או כאשר כל המתגים נמצאים בOff.

אם תצליחו להדליק את הנורה בעשרה מהלכים, תשוחררו. אם לא, הפסיכופת יקבל אקדח ויעשה איתו מה שהוא רוצה.

בכל מהלך מישהו מנתק מבחוץ את המפסק הראשי, ואתם יכולים ללחוץ על כל מתג שאתם רוצים (אחד או יותר). לאחר מכן, המפעיל מחבר את המפסק הראשי כדי לראות אם הפעולה הצליחה (כלומר, האם הנורה נדלקה). אם לא, הפסיכופת מסובב את השולחן באופן אקראי (אין לכם אפשרות לעקוב אחרי הסיבוב של השולחן). השולחן והמתגים סימטריים לחלוטין, ולכן אתם לא יכולים לדעת על אילו מתגים לחצתם קודם.

המטרה: הגדירו רצף של **פעולות** (רמז: אילו פעולות מוכרות לכם מהמסמך הזה?) שיחלץ אתכם מהחדר וישאיר שם את הפסיכופת לבד עם האקדח שיתנו לו.

בהצלחה!