תרגיל 5

הנחיות הגשה

1. העבודה היא ביחידים.
2. ההגשה היא עד ליום חמישי, בתאריך 13.1.2022 בשעה 23:30.
3. יש להגיש את הקובץ myfunction.c בלבד. הגשת התרגיל תתבצע באמצעות submit.
4. בשורה הראשונה של הקובץ אותו אתם מגישים, יש לרשום בהערה מס' ת.ז. ושם מלא. לדוג':

// 123456789 Israela Israeli

רקע כללי

בתרגיל זה נתון לכם קטע קוד תקין (נכון לוגית, ללא באגים) שלא עבר אופטימיזציה, ומטרתכם היא לייעל את זמן הריצה שלו ככל הניתן בעזרת כל הכלים והעקרונות אשר למדתם בקורס. המטרה היא להגיע לזמן ריצה מהיר ככל האפשר של התוכנית, תוך שמירת נכונות לוגית. הניקוד עבור תרגיל זה יינתן על בסיס תחרות בביצועים, כך שזמן ריצה קצר יותר ייקבל ניקוד גבוה יותר.

<u>שימו לב</u>: הקובץ היחיד אשר אתם רשאים לשנות הוא myfunction.c. ניתן לבצע בו כל שינוי, כולל מחיקה והוספה של פונקציות, שינוי אלגוריתמי, שימוש ב- magic numbers וכו', כל עוד שומרים על פלט התוכנית זהה למקור.

התקנת סביבת העבודה

בשורת הפקודה יש לרשום את הפקודה הבאה:

```
sudo apt-get install freeglut3-dev libxmu-dev libxi-dev
```

הקוד

בקובץ myfunction.c, הפונקציה myfunction מקבלת תמונה וערך בחירה נוסף.

1. אם הערך שמתקבל הוא 1, הפונקציה מבצעת החלקה ("טשטוש") של התמונה ולאחר מכן מבצעת חידוד של התמונה המוחלקת. התמונה לאחר ההחלקה וגם התמונה לאחר החידוד נשמרות בתור קבצי bmp. זה מתקבל ע"י קריאה לפונקציה writeBMP שבקובץ writeBMP.c (קוד שאותו לא משנים).
2. אם מתקבל ערך שונה מ- 1, הפונקציה מבצעת החלקה **<u>עם פילטר</u>** של התמונה ולאחר מכן מבצעת חידוד של התמונה המוחלקת.
3. במידה ולא יתקבל ערך נוסף לאחר התמונה, תיזרק הודעת שגיאה ותסיים את התוכנית.

ייצוג תמונה במחשב

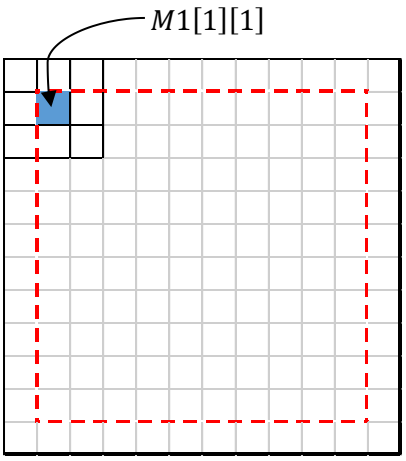
תמונה במחשב מיוצגת כמטריצה, שבה בכל תא שמור ערך הפיקסל המתאים. דרך נפוצה לייצוג תמונה צבעונית במחשב היא שיטת RGB:
בכל תא שומרים 3 ערכים: Red, Green, Blue (ייצוג RGB). כלומר, כל צבע מיוצג ע"י שילוב של 3 צבעים- אדום, ירוק וכחול (נקראים גם "ערוצים").

החלקה וחידוד של תמונה (פעולת קונבולוציה)

פעולת "החלקה" (smooth) של תמונה היא החלפת כל פיקסל בממוצע של כל הפיקסלים שמסביבו בתוך "חלון" בגודל 3X3 שבמרכזו נמצא הפיקסל. פעולת "החלקה" עבור תמונת RGB היא ביצוע החלקה עבור ערכי R בנפרד, ערכי G בנפרד וערכי B בנפרד.

למשל, עבור ערוץ מסוים (אחד מ-R,G או B), בהינתן תמונה M1 בגודל NXN, הפיקסל במקום ה-[1,1] יקבל את הערך הבא:

1. $M2[1][1] = \frac{\sum_{i=0}^{2}\sum_{j=0}^{2} M1[i][j]}{9}$



כאשר M2 זו התמונה לאחר ה"החלקה". שימו לב שערכי הפיקסלים השכנים נלקחים מהתמונה **המקורית**.

נרצה להתייחס אך ורק לפיקסלים שאפשר "להניח" עליהם את החלון, כלומר פיקסלים שיכולים להימצא במרכז חלון שכזה בגודל 3X3. פיקסלים אלו הם הפיקסלים שבתוך התחום המסומן באדום.

כלומר, אנחנו מתייחסים אך ורק לפיקסלים עם אינדקס שורה i המקיים $\frac{kSize}{2} \le i < N - \frac{kSize}{2}$ ועם אינדקס עמודה j המקיים $\frac{kSize}{2} \le j < N - \frac{kSize}{2}$, כאשר $kSize$ הוא גודל החלון (במקרה שלנו, $kSize = 3$).

לאחר סיום המעבר עם החלון על התמונה המקורית, התמונה המתקבלת נראית יותר מטושטשת, ולכן הפעולה נקראת "החלקה".

ניקח מטריצה K בגודל 3x3 (נקרא לה "מסיכה", או kernel) ונעביר אותה על הפיקסלים של התמונה, כך שבכל פעם פיקסל אחר של התמונה המקורית יהיה במרכז המסיכה – כמו הפיקסל $M1[1][1]$ באיור הנ"ל.

אם K היא מהצורה:

$$K = \frac{1}{9} \cdot \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \end{bmatrix}$$

אז נשים לב שנוסחא 1 הנ"ל שקולה לנוסחא הבאה:

$$M2[1][1] = \sum_{i=0}^{2}\sum_{j=0}^{2} M1[i][j] \cdot K[i][j]$$

למעשה, אפשר להכליל את הפעולה הנ"ל ולא להשתמש אך ורק בחישוב ממוצע, אלא לשנות את הערכים ב-K (נקראים גם "משקלים") ובכך לבצע סכום ממושקל אחר עבור הפיקסל שיהיה במרכז המסיכה. כך נוכל פעולות שונות על תמונה נתונה.
למשל, עבור K מהצורה

$$K = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

מתקבלת תמונה "חדה" יותר.
העברת מסיכה על תמונה וחישוב הסכום הממושקל (בהתאם לערכים שבמסיכה) עבור כל פיקסל, נקראת פעולת קונבולוציה.

קימפול הקוד

לתוכנית מצורף makefile. כמובן שאין צורך לשנות אותו מכיוון שבבדיקה ייעשה שימוש ב-makefile המקורי. כרגיל, כדי למקפל את התוכנית יש להשתמש בפקודה make מהטרמינל.

הרצת התוכנית

לאחר קימפול תקין של התוכנית נוצר קובץ ריצה בשם showBMP. הרצת התוכנית מתבצעת בצורה הבאה:

```
./showBMP <imageName> <kernelOption>
```

לדוגמא:

```
./showBMP gibson_500.bmp 1
```

או

```
./showBMP gibson_500.bmp 2
```

פקודות אלו יריצו את התוכנית עבור קובץ התמונה gibson_500.bmp אשר התקבל כקלט הראשון ויפעילו פילטר או לא בהתאם לערך הנוסף שהתקבל כקלט השני.

שימו לב:

אתם יכולים להניח שקובץ התמונה שיתקבל ל-myfunction תמיד יכיל תמונה ריבועית (כלומר, מגודל $n \times n$) אך הקוד שלכם יבדק גם על קבצי תמונה בעלי n שונה (!)

בפרט, ניתן להניח כי $100 \leq n \leq 1000$.

מדידת ביצועי התוכנית

השינויים שתבצעו בקוד ישפיעו על האופן בו מתבצעת הפעולה, אבל לא על התוצאה שלה – **שימו לב שהקוד שלכם והקוד המקורי יוצרים את אותה התמונה**.
מעבר להתבוננות בשתי התמונות, תוכלו להיעזר בעובדה שהרצת התוכנית כותבת לקובץ את התמונות שנוצרות. בתיקיית הקבצים הנלווים לתרגיל (ex5_files.zip שבאתר) תוכלו למצוא את קבצי התמונה שנוצרו עבור הפעלת הקוד המקורי של myfunction: Blur_correct.bmp עבור התמונה המוחלקת ו-Sharpen_correct.bmp עבור התמונה לאחר החידוד. באותו אופן Filtered_Blur_correct ו- Filtered_Sharpen_correct עבור המקרים שהופעל פילטר. היעזרו בפקודה cmp כדי להשוות בין הקבצים הללו לבין הקבצים שיצר הקוד שלכם.

על מנת לבחון את הביצועים שלכם על תמונות נוספות העומדות בקריטריונים, ניתן להיעזר בחיפוש תמונות באינטרנט, למשל באמצעות https://images.google.com/ וחיפוש לפי imagesize:WIDTHxHEIGHT (לדוגמה imagesize:200x200). לאחר הורדת תמונה למחשב שלכם ניתן להיעזר בכלי המרה מכל סוג תמונה לסוג הדרוש לנו (bmp), למשל באמצעות העלאת התמונה ל- https://online-converting.com/image/convert2bmp/ ושמירה מחדש לאחר שההמרה מתבצעת. שימו לב שבאתר הזה ברירת המחדל היא לשמור את התמונה החדשה תוך שימוש בהגדרה image depth = 24 bit שזה מה שאנחנו צריכים (שכן יש שלושה צבעים וכל צבע מתואר ע"י 8 ביט), ואת זה ניתן לאמת לפי האינדיקציה הבאה:

Color: 24 (True color)

לבסוף, חשוב לציין שזמן הריצה של myFunction עשוי להשתנות גם בין שתי הרצות של אותה התוכנית בדיוק על אותו הקלט, וזה קורה בין היתר בשל זמני כתיבה משתנים לקובץ (שהרי myFunction יוצרת שתי תמונות חדשות). כדי לוודא שאתם באמת מצליחים לחסוך בזמן ריצה, מומלץ לחשב ממוצע עבור מספר ריצות זהות (לפחות 10) כדי לקבל תוצאה אמינה.

הגשת התוכנית

יש להגיש את הקובץ myfunction.c בלבד. אין צורך להגיש את שאר הקבצים מכיוון שהקוד שלכם יקומפל עם תוכנית הבדיקה.

ציון התרגיל

כלל הבסיס הוא שהתוכנית החדשה חייבת לספק פלט זהה למקור, אחרת תתבצע הורדה משמעותית של נקודות. בהינתן פלט תקין, הפרמטר העיקרי לקביעת הציון במטלה זו הוא זמן הריצה שלכם, כך שזמן ריצה קצר יותר יזכה לציון גבוה יותר. בנוסף, יצירתיות ו/או מאמץ יוצא דופן עשויים לזכות בנקודות נוספות.

הערות

1. בתרגיל זה ייבדקו קלטים בגדלים שונים מהקלט הנתון, כאמור $100 \leq n \leq 1000$.
2. יש צורך בהערות משמעותיות (להסביר כיצד נעשה השיפור לקוד). רלוונטי עבור ערעורים.
3. התרגילים ייבדקו על planet. הגשת התרגיל תחזיר פידבק שכולל זמני ריצה של 20 הרצות של התוכנית שלכם על התמונה המצורפת ב-ex5_files.zip. **התעלמו מהציון שיתלווה לפידבק, הוא לא מספק לכם מידע השוואתי לביצועים של סטודנטים אחרים.**

בהצלחה!