תרגיל בית 3

הנחיות:

1. תאריך הגשה: 26.1.2023 בשעה 23:50
2. **הגשה בזוגות בלבד.**
יש להירשם כקבוצה במודל לפני ההגשה.
רק אחד משני השותפים בקבוצה צריך להגיש את התרגיל.
3. **הורידו את קבצי השלד מהמודל וממשו את הפונקציות החסרות בהם.**
4. במודל נמצא לינק לאתר בו מופיעות הנחיות כתיבת קוד בפורמט של פרויקט GNU.
בנוסף מצורף קובץ PDF עם סיכום הנחיות הקידוד בעברית.
הקוד צריך להיכתב בהתאם למסמך ההנחיות. אנא קראו אותו בקפידה.
הגשה שלא תעמוד בהנחיות תגרור הורדת ניקוד.
5. הגשה של קובץ zip בודד. שם הקובץ בפורמט הבא:

HW3_ID1_ID2.zip

כאשר ID1 ו-ID2 הם תעודות הזהות של המגישים.
6. בתוך קובץ ה-zip יש להגיש את הקבצים:

bmp.h
filter.c
helpers.c
helpers.h

ומלאו את תעודות הזהות שלכם כהערה במקום המתאים ב-helpers.c.
7. הוסיפו הערות המסבירות את הקוד שלכם. קוד ללא הערות לא יזכה בניקוד מלא.
8. אסור להוסיף לייבא ספריות נוספות.
9. וודאו כי הקוד שלכם מתקמפל. קוד שאינו עובר קומפילציה יגרור הורדה משמעותית בציון.
10. בדקו את הקוד שלכם עם קלטים שונים ע"מ לבחון מקרים נוספים ומקרי קצה.
ע"מ לבצע בדיקות נוספות, עקבו אחר ההוראות הנמצאות ב**נספח 1**.
שימו לב: מותר לכם לשנות את התוכן של הקובץ **helpers.c** בלבד. אין לגעת בתוכן של אף אחד מהקבצים האחרים.
11. במקביל לפתיחת התרגיל נפתח פורום שאלות במודל.
שאלות לגבי התרגיל ניתן לשאול שם.

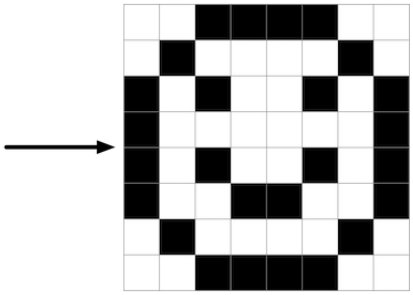
משימה 1

בתרגיל זה תכתבו תוכנית המבצעת מניפולציה (פילטר) על תמונה.

רקע

ניתן לחשוב על תמונה כעל מטריצה, אשר בכל תא במטריצה ניתן לבחור צבע מסוים.
בשביל לייצג תמונה בשחור-לבן, ניתן לשים בתאים המתאימים 1 לתא לבן ו-0 לתא שחור.
התבוננו בדוגמא הבאה:

```
1 1 0 0 0 0 1 1
1 0 1 1 1 1 0 1
0 1 0 1 1 0 1 0
0 1 1 1 1 1 1 0
0 1 0 1 1 0 1 0
0 1 1 0 0 1 1 0
1 0 1 1 1 1 0 1
1 1 0 0 0 0 1 1
```



למעשה, כל תא במטריצה שלנו הוא פיקסל.
בשביל לייצג תמונה בשחור לבן, אנו זקוקים לביט אחד בלבד בכל תא (ביט יכול לשמור את הערכים 0 או 1).
בשביל ליצור צבעים נוספים, נצטרך לשמור בכל תא מידע נוסף בעזרתו נוכל לייצג את הצבע של הפיקסל הנוכחי.
פורמטי תמונות כמו BMP, JPEG ו-PNG תומכים בדיוק באפשרות זו. פורמט BMP מאפשר לנו לייצג תמונות בעזרת 24 ביטים לפיקסל (למעשה הוא תומך גם בגדלים נוספים אך אנו נשתמש בפורמט 24 ביטים בלבד).

ב-24-BIT BMP כל אחד מהצבעים אדום, ירוק וכחול (RGB) מיוצג באמצעות 8 ביטים.
נזכיר, כי 8 ביטים מייצגים מספר בין 0-255. המספר עבור כל אחד מהצבעים מייצג עד כמה הוא יהיה בולט בתמונה המקורית.
למשל פיקסל שבו ערך האדום הוא 0xff, ערך הירוק הוא 0x00, וערך הכחול הוא 0x00 יופיע כפיקסל אדום לחלוטין.
בפורמט BMP סדר הצבעים נשמר הפוך עבור כל פיקסל – כלומר קודם כל מופיע ערך הצבע הכחול, לאחר מכן הירוק ואז האדום.
אם היינו רוצים לייצג את התמונה הקודמת בפיקסלים מסוג 24-BIT, כאשר במקום צבע שחור נשתמש בצבע אדום – נקבל את הייצוג הבא:

```
ffffff ffffff 0000ff 0000ff 0000ff 0000ff ffffff ffffff
ffffff 0000ff ffffff ffffff ffffff ffffff 0000ff ffffff
0000ff ffffff 0000ff ffffff ffffff 0000ff ffffff 0000ff
0000ff ffffff ffffff ffffff ffffff ffffff ffffff 0000ff
0000ff ffffff 0000ff ffffff ffffff 0000ff ffffff 0000ff
0000ff ffffff ffffff 0000ff 0000ff ffffff ffffff 0000ff
ffffff 0000ff ffffff ffffff ffffff ffffff 0000ff ffffff
ffffff ffffff 0000ff 0000ff 0000ff 0000ff ffffff ffffff
```

קובץ BMP מכיל גם מידע על התמונה עצמה (metadata), כמו הגובה והרוחב שלה, המשקל של הקובץ ומידע נוסף אשר מופיע בתחילת הקובץ ואינו מייצג פיקסלים.
ישנם שני חלקים למידע זה, והם לוקחים יחד בדיוק 54 bytes. מיד לאחר מכן נמצאים הפיקסלים שמרכיבים את התמונה.
ניתן לחשוב על התמונה כעל מערך דו-ממדי של פיקסלים, וכך בדיוק תוכלו לגשת לכל פיקסל בתמונה בתרגיל זה.

בתרגיל הקרוב תצטרכו להטמיע פילטרים שונים על תמונות.

Grayscale – המרת תמונה מצבע לתמונה בגווני אפור

כיצד ניתן להמיר תמונה בצבע תמונה ב"שחור-לבן"?
פיקסל שבו הערך של כל אחד מהצבעים הוא 0x00 (כלומר 0 ב-hexadecmial) יופיע כפיקסל שחור בתמונה, ובפיקסל לבן ערך כל אחד מהצבעים הוא 0xff (255 ב-hexadecimal). פיקסל יופיע בגוון של אפור כאשר ערך הצבע האדום, הירוק והכחול שווים – ככל שהערך שלהם קטן יותר – כך הפיקסל יופיע כהה יותר וקרוב יותר לשחור.
בשביל להמיר תמונה בצבע לתמונה בגווני אפור, נצטרך לדאוג שערכי הצבעים בפיקסל יהיו אותו הערך. אחת מהשיטות לבחור את הערך הזה הוא לחשב את הממוצע של הערכים של שלושת הצבעים ולהשתמש בו.

Sepia

אתם בוודאי מכירים מתוכנות עריכה רבות את הפילטר הנותן לתמונה גוון אפור-חום.

ישנן דרכים רבות לממש את הפילטר הזה, בתרגיל הנוכחי, תשתמשו באלגוריתם הבא המחשב עבור כל פיקסל את הצבע שלו בשביל פילטר ספיה בהתאם לערכי הפיקסל המקוריים:

```
sepiaRed = .393 * originalRed + .769 * originalGreen + .189 * originalBlue
sepiaGreen = .349 * originalRed + .686 * originalGreen + .168 * originalBlue
sepiaBlue = .272 * originalRed + .534 * originalGreen + .131 * originalBlue
```

סביר כי התוצאה של הפיקסל החדש אינה מספר שלם, ולכן תצטרכו לעגל אותה, לשם כך מצאו פונקציה מתאימה בתוך ספריית math.h.
אם ערך צבע מסוים עבר את הרף העליון, כלומר עבר את המספר 255, עליכם לתקן אותו חזרה לערך 255.

Reflect/Mirror – שיקוף התמונה בציר x

לעיתים נרצה לקחת תמונה וליצור מחדש את התמונה כאילו שמנו אותה מול מראה.
כיצד ניתן לעשות זאת? חשבו על כיצד משתנה המיקום של כל פיקסל בתמונה.
האם הגובה שלו משתנה? האם השורה בה הוא נמצא משתנה? אם כן, איך מתבצע השינוי?

<u>הקבצים בתרגיל:</u>

bmp.h - התבוננו בקובץ bmp.h. קובץ זה מגדיר את המבנים שקובץ BMP משתמש בהם.
מדוע הקובץ מגדיר מבנים (struct)? קובץ הוא פשוט אוסף של נתונים בינאריים. אבל מאוד קשה לעבוד עם מערך ארוך של נתונים בינאריים – לכן אנו מגדירים מבנים (struct). בעזרת struct, אנו יכולים לתת שמות לאזורים שונים בקובץ ולגשת אליהם יותר בקלות. כעת ניתן לחשוב על קבצים כעל סדרה של struct מסודרים אחד אחרי השני – ולא כעל מערך ענק של ביטים.
בקובץ ניתן לראות את השדות עליהם דיברנו קודם, כלומר קובץ BMP נראה כך:

הסבר	גודל	שם השדה
מידע טכני על התמונה	14 bytes	BITMAPFILEHEADER
מידע טכני על התמונה	40 bytes	BITMAPINFOHEADER
מבנה של פיקסל בודד	8 bits + 8 bits + 8 bits = 24 bits = 3 bytes	RGBTRIPLE

כאשר ניקח מערך של פיקסלים שגודלו הוא גובה התמונה * רוחב התמונה – נקבל את כל הפיקסלים של התמונה.

filter.c – קובץ זה מגדיר את עיקר אופן פעולת התוכנה שלנו. הוא מקבל את שם התמונה אותה צריך לקרוא מהמשתמש, פותח את התמונה, טוען אותה לזכרון, מחלץ מידע רלוונטי לגביה – כמו הגובה והרוחב שלה, טוען את מערך הפיקסלים של התמונה, מפענח איזה פילטר המשתמש ביקש להפעיל על התמונה, ובסוף כותב את קובץ התמונה בשם שנתן המשתמש.
כדאי לעבור על הקובץ ולהבין בגדול מה הוא עושה – ישנן הערות לכל אורך הקובץ המסבירות מה קורה בכל שורה.

helpers.h – קובץ המגדיר את הפעולות אותן תצטרכו לממש בתרגיל זה. שימו לב, בקובץ זה נמצאות רק ההכרזות על הפעולות, ולא המימוש שלהן.

helpers.c – בקובץ זה תממשו את הפונקציות המבצעות את השינוי של התמונה בהתאם לפילטר הנבחר.
כל אחת מהפונקציות מקבלת פרמטרים זהים:

```
int height, int width, RGBTRIPLE image[height][width]
```

`height` – מכיל את הגובה של התמונה
`width` – מכיל את רוחב התמונה
`RGBTRIPLE image[height][width]` – מערך דו ממדי בגודל `height×width` פיקסלים בטיפוס `RGBTRIPLE`.

בקובץ זה יש דוגמא לפילטר שאנו מימשנו בשביל להדגים כיצד לשנות פיקסלים של תמונה. המבנה והתוכן של הפונקציה `blacksquare` יכול לעזור לכם להתחיל לפתור את התרגיל.

הנחיות לפתרון:
בקבצי התרגיל קיבלתם 4 קבצי קוד – filter.c, helpers.c, helpers.h, bmp.h.
את הקוד שלכם תממשו בתוך הקובץ helpers.c בלבד, אסור לשנות את שאר הקבצים.

עליכם לממש את הפונקציות הבאות:

```
void grayscale(int height, int width, RGBTRIPLE image[height][width])
void sepia(int height, int width, RGBTRIPLE image[height][width])
void reflect(int height, int width, RGBTRIPLE image[height][width])
```

בנוסף, ישנה תיקיית images עם תמונות שתוכלו להשתמש בשביל לבדוק את הפונקציות שלכם.
כמו תיקיית filter_examples עם דוגמאות לפלט – כלומר תמונות עם הפילטרים השונים.

בהצלחה!

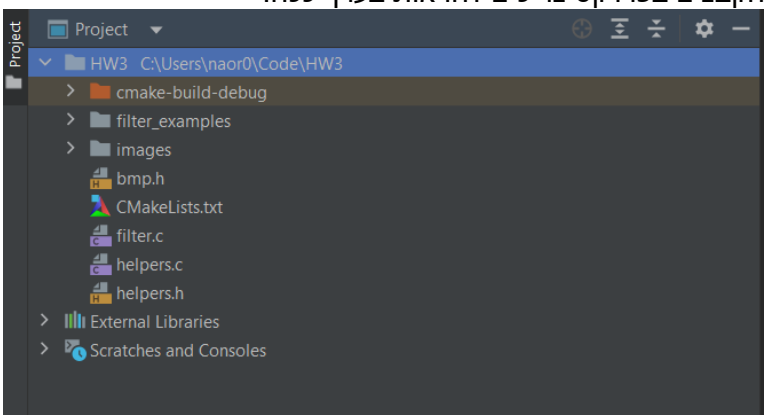
נספח 1

הנחיות להרצה:

במידה והאופציות שמופיעות בהוראות אינן תואמות את האופציות המופיעות אצלכם במחשב, הכנסו ללינק הזה והתקינו את MinGW לפי ההוראות המפורטות, לאחר מכן סגרו ופתחו מחדש את Clion.

פתחו פרויקט חדש ב-Clion וחלצו את הזיפ HW3.zip מהמודל לתוך תיקיית הפרויקט.

הקבצים בפרויקט צריכים להראות בערך ככה:



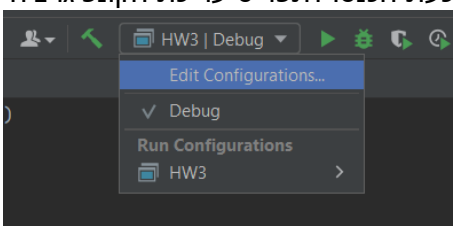
פתחו את הקובץ CMakeLists.txt, ומצאו את השורה בה כתוב:

```
add_executable(HW3 main.c)
```

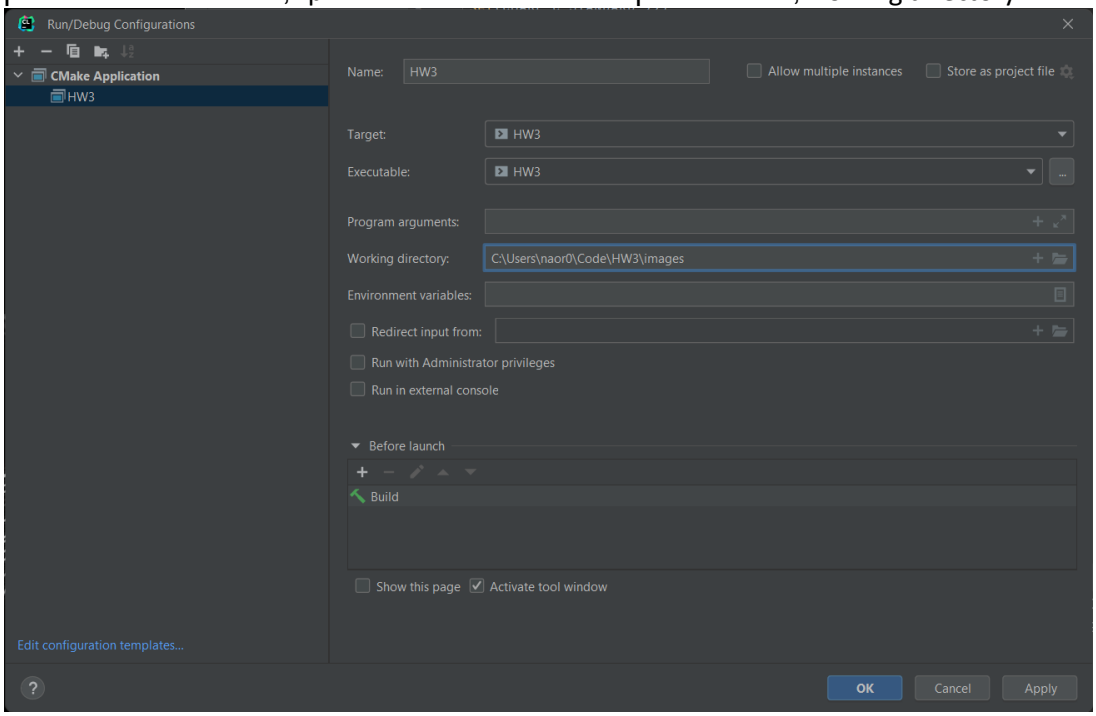
מחקו את main.c והוסיפו את filter.c ואת helpers.c. השורה צריכה להראות כעת כך:

```
add_executable(HW3 filter.c helpers.c)
```

כעת הכנסו לתפריט עריכת הקונפיגרציה המופיע בצד ימין של המסך:



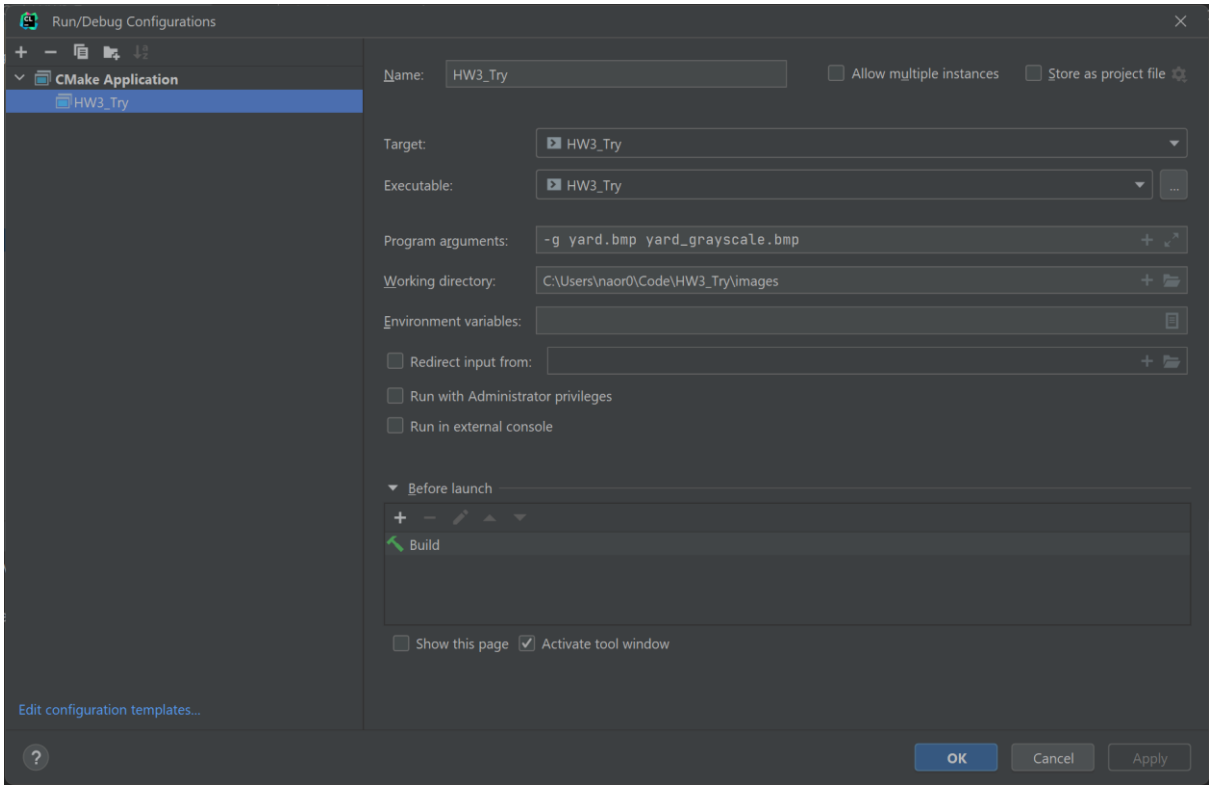
בשדה Working directory, בחרו את תיקיית התמונות שחילצתם מה-zip, במחשב שלי זה נראה כך:



דך שדה ה-Program Arguments תוכלו לבחור את התמונה והפילטר אותו אתם מפעילים.
תצטרכו להכניס את הפילטר, שם התמונה שאתם מכניסים (אחד מהקבצים בתיקייה images) ואת שם הקובץ אותו התוכנה תיצור בשדה זה.
בשביל לבחור את הפילטר השתמשו באות הראשונה של הפילטר עם מקף לפניה:
b- בשביל הדוגמא שאנו מימשנו
g- בשביל גווני אפור
r- בשביל שיקוף
s- בשביל ספיה (sepia)

לדוגמא, בשביל ליצור להפעיל את הפילטר גווני אפור על תמונה בשם yard.bmp וליצור קובץ בשם yard_grayscale.bmp נכתוב את הטקסט הבא בשדה הProgram Arguments:
-g yard.bmp yard_grayscale.bmp

מסך הקונפיגרציה ייראה כך:



לחצו על OK בשביל לשמור את האופציות שבחרתם.
לאחר שתקמפלו ותריצו את הקוד שלכם (לחיצה על המשולש הירוק), יהיה קובץ חדש בתיקייה images עם השם שנתתם, במקרה זה yard_graysacle.bmp.
אם תכנסו מחדש למסך הקונפיגורציה, תשנו את הפילטר ל-b, תשנו את שם התמונה היוצאת (למשל yard_blacksquare.bmp) ותריצו את התוכנית, תוכלו לראות את הקובץ החדש שנוצר עם פילטר הדוגמא שמימשנו בשבילכם.