תרגיל בית 3 (שפת ++C)

הנחיות חשובות:

- בתרגיל זה מותר לכם להשתמש בקבצים שלכם בספריות הסטנדרטיות שמוכללות כבר בקובץ GrayvalueImage.hpp אין להכליל ספריות נוספות
- בתרגיל זה אסור להשתמש ב new ו- delete או בהקצאה דינמית אחרה באופן ישיר.
- בתרגיל זה אין להשתמש במילה השמורה using (שימוש בקובץ header נחשב סגנון גרוע)
- שימו לב שלקבצים מצורפים קבצי קוד וההוראות בקבצים הללו מחייבות!

הנחיות חשובות לכלל התרגילים מעתה והלאה בקורס:

- התרגילים הם לעבודה ביחידים. מותר להתייעץ אך ורק בעל פה, אסור בתכלית האיסור שחומר כתוב/מודפס/אלקטרוני יעבור בין אנשים. בנוסף, על חלק מהתרגילים תיבחנו פרונטלית ועליכם להבין כל דבר בקוד!
- המנעו ממספרי קסם: מספרים שמופיעים באמצע הקוד בלי משמעות מיוחדת (לדוגמא נניח שמספר הרשומות בתרגיל אחר הוא מקסימום 50 ואז בכל מקום בקוד כתוב 50. לעומת זאת, 0 לתחילת מערך לא נחשב מספר קסם - הפעילו הגיון בריא) והשתמשו במקום זאת בפקודות מאקרו (#define או אם כבר למדתם על כך ב const).
- אין להשתמש ב-variable length arrays , וכן במשתנים סטאטיים ו / או גלובליים.כל התרגילים בקורס צריכים להתקמפל ולרוץ באתר c9.io (האתר מריץ מערכת הפעלה אובונטו) עם ה Makefile המצורף
- יש להקפיד על סגנון תכנות טוב כמו שלמדתם. לדוגמא, להימנע מחזרות קוד (לכתוב פונקציות שצריך), שמות משתנים עם משמעות, בהירות הקוד, תיעוד הקוד, להקפיד להשתמש בקבועים שצריך ולא במספרי קסם וכו'.
- <u>אופן כתיבת ההערות</u>: בכל תחילת קובץ הצהרות (h. או hpp.) יש לכתוב את תפקיד הקוד שבקובץ.
  כמו-כן לפני כל פונקציה ומתודה ומשתנה מחלקה/מבנה יש לכתוב הערה על תפקידם.
  יש להוסיף הערות במימוש לפי הצורך.
  בקורס זה במידה וניתנו לכם הוראות מפורטות לגבי פונקציה בתרגיל מותר לכתוב ראה בהגדרת התרגיל במקום לעשות copy&paste.
- עליכם להגיש קובץ ששמו מספר ת.ז. שלכם כמו שהיא מופיעה באתר המודל נקודה zip.
  לדוגמא, אם מספר ת.ז. שלי הוא 12345678 אז שם הקובץ יהיה:

12345678.zip

חובה להשתמש ב make zipfile ע"מ לייצר קובץ תקין ו make checkzipfile ע"מ לבדוק אותו
חשוב: חלק מהבדיקה אוטומטית ואי עמידה בקמפול/בדיקה ע"י ה Makefile הניתן יגרור אפס מיידית.

- הקפידו להשתמש בפונקציות של ++C (למשל new, delete, cout) על פני פונקציות של C (למשל malloc, free, printf). בפרט השתמשו במחלקה string ולא במחרוזת של C (כלומר, * char).

- יש להשתמש בספריות סטנדרטיות של ++C ולא של C אלא אם כן זה הכרחי.
- הקפידו על עקרונות Information hiding. לדוגמא, הקפידו להגדיר משתני מחלקות כ private.
- הקפידו לא להעביר אובייקטים גדולים by value אלא by reference או by const reference.
- הקפידו מאד על const correctness. כלומר, על שימוש במילה השמורה const בצורה נכונה.
- שאלות על התרגיל יש לשאול בפורום המתאים במודל בלבד!

<u>תרגיל:</u>

נתונה לכם מחלקה שמייצגת תמונה שחורה לבנה. לתמונה מספר שורות, מספר עמודות, דרגת אפור/לבן מקסימלית (מייצגת לבן) ומערך לא פרימיטיבי (vector) שמכיל את כל הפיקסלים של התמונה שורה אחר שורה. אתם יכולים להשתמש בפונקציה getPixel על מנת לקבל את ערך הפיקסל בשורה וטור (שימו לב שהספירה מתחילה כרגיל ב-0).

המשימות שלכן הן:

1. להפוך את המחלקה למחלקה גנרית כאשר טיפוס דרגת האפור יהיה גנרי וברירת המחדל שלו תהיה unsigned char. כלומר, אם כותבים:
```
GrayvalueImage<> im;
```
מתבצעת קריאה לבנאי ברירת מחדל של תמונה שטיפוס דרגת האפור שלה הוא unsgined char.
2. לממש את פונקצית פילטר החציון. ראו פירוט בהמשך.
3. לממש את אופרטור ההשוואה == לפי התיעוד שלו
4. לממש את אופרטור ההשוואה =! לפי התיעוד שלו
5. לממש את swap_rows לפי התיעוד שלו. שימו לב שנדרש זמן ריצה של O(1) שימו לב שכרגיל כל שאר הפונקציות צריכות להמשיך לעבוד טוב לאחר הפעולה!
6. לממש את swap_cols לפי התיעוד שלו. שימו לב שנדרש זמן ריצה של O(1) שימו לב שכרגיל כל שאר הפונקציות צריכות להמשיך לעבוד טוב לאחר הפעולה!
7. להשתמש בטיפוס size_t לאינדקסים. כמו שראינו זה הוא טיפוס שלא יכול לאחסן ערכים שליליים והוא גולש למספר מאד גדול אם מנסים להכניס לו 1- ולכן יש להימנע מכך. מי שימיר אינדקסים ל int במהלך התוכנית יורדו לו 2 נקודות.
8. להבין לעומק את כל הקוד גם זה שכתבתם וגם זה שניתן לכם! במבחן הפרונטלי תיצטרכו להראות הבנה של כל הקוד כולל סיבוכיות זמן הריצה של כל פעולה!

מה מותר לכם לעשות:

1. להוסיף שדות ופונקציות פרטיות.
2. לשנות את המימוש של חלק מהפונקציות שניתנו לכם.

מה אסור לכם לעשות:

1. לשנות את החתימה של הפונקציות. לדוגמא:

```
void medianFilter(size_t radius, unsigned int method)
```

<u>מימוש median filtering לתמונה שחור לבן (דרגות אפור).</u>

בסעיף זה אתם צריכים לממש פונקציה אחת (מומלץ לכתוב פונקציות עזר נוספות):

```
void medianFilter(size_t radius, unsigned int method);
```

פונקציה זאת מחליפה כל ערך פיקסל בערך החציון בסביבת הריבוע בגודל (2*radius+1)x(2*radius+1) אשר נמצא סביבו.

החציון בתרגיל זה מוגדר כערך במיקום **(num_of_neighborhood_pixels-1)/2** במערך הממויין של כל ערכי הפיקסלים החוקיים בריבוע מסביבו כאשר num_of_neighborhood_pixels הוא מספר הפיקסלים החוקיים בריבוע מסביבו. פיקסל "לא חוקי" הוא פיקסל אשר אינו נמצא בגבולות התמונה. שימו לב שבתרגיל זה אנו פשוט מתעלמים מפיקסלים כאלו.

מוגדרים לכם 3 שיטות שאתם צריכים לממש למציאת החציון לכל פיקסל:

1. שיטה שמשתמשת ב std::sort
2. שיטה שמשתמשת ב std::nth_element
3. שיטה שמשתמשת בהיסטוגרמות. פירוט נוסף בהמשך.

שימו לב שאתם לא יכולים לעדכן כל פיקסל במערך המקורי אלא אתם חייבים לעדכן את הפיקסלים במערך חדש או לחלופין לקרוא את הערך של השכנים מעותק של המערך הישן לפני העדכון. זאת על מנת שלא לעדכן ערך של פיקסל לפי שכנים שעודכנו כבר.

יש לקרוא את התיעוד של הפונקציה ולהבין אותה במלואה!

הקובץ המצורף חייב להתקמפל עם g++ באובונטו (אפשר לוודא ב c9io) כאשר מריצים ב shell את הפקודה make עם הקובץ Makefile הנתון.

מומלץ מאד גם לכתוב תוכנית בדיקה ע"י גוגל טסט (להשלים את הקובץ GrayvalueImageTest.cpp). אין צורך להגיש את הקובץ GrayvalueImageTest.cpp

<u>שיטת ההיסטוגרמות:</u>

שיטת ההיסטוגרמות משתמשת ב-2 מערכי עזר:

1. מערך אחד באורך של מספר דרגות האפור (MAX_G_ + 1). מערך זה מכיל ספירה (היסטוגרמה) עבור כל ערך אפשרי של דרגת אפור כמה פיקסלים יש בריבוע בגודל

(2*radius+1)x(2*radius+1)

שמקיף אותו בתמונה (כולל הוא עצמו). ממערך זה אפשר למצוא את החציון בזמן ריצה:

O(MAX_G)

לדוגמא, כאשר אנו מחשבים את החציון עבור הפיקסל בשורה השלישית ובטור הרביעי אלו הפיקסלים שנתייחס אליהם

		1	5	3			
		2	8	5			
		5	5	4			

ובהנחה שהערך המקסימלי של דרגות האפור הוא 9 המערך יהיה:

0, 1, 1, 1, 1, 4, 0, 0, 1, 0

2. מערך שמכיל עבור כל טור מערך באורך של מספר דרגות האפור (MAX_G_ + 1) שכל מערך כזה מכיל ספירה (היסטוגרמה) עבור כל ערך אפשרי של דרגת אפור כמה פיקסלים יש בטור בגודל

(2*radius+1)

שמעליו ומתחתיו.

לדוגמא, כאשר אנו מחשבים את החציון עבור פיקסלים בשורה הראשונה אלו הפיקסלים שנתייחס אליהם כאן:

1	5	3	2	1
2	8	5	2	2

כאשר המערך של המערכים הזה (שוב בהנחה שהערך המקסימלי של דרגות האפור הוא 9):

0 0 0 0 0
1 0 0 0 1
1 0 0 2 1
0 0 1 0 0
0 0 0 0 0
0 1 1 0 0
0 0 0 0 0
0 0 0 0 0
0 1 0 0 0
0 0 0 0 0

איתחול מערכי העזר הללו יכול להתבצע בזמן ריצה של

O(C*MAX_G+C*radius)

לאחר שמאתחלים את המערכים הללו אנו נעבור על כל הפיקסלים ונחשב מהמערך הראשון את החציון בזמן ריצה:

O(MAX_G)

מכיוון שאנו עושים זאת עבור כל פיקסל סה"כ זמן הריצה של כל החישובים הנ"ל הוא:

O(R*C*MAX_G)

כל פעם שעוברים טור נעדכן את המערך הראשון בעזרת המערך השני ע"י כך שנוריד את ההיסטוגרמה של הטור השמאלי הישן (ממערך 2) ונוסיף את ההיסטוגרמה של הטור הימני החדש (ממערך 2). עדכון זה יתבצע גם כן בזמן ריצה:

O(MAX_G)

מכיוון שאנו עושים זאת עבור כל פיקסל סה"כ זמן הריצה של כל החישובים הנ"ל הוא:

O(R*C*MAX_G)

מה שנשאר לעשות הוא לעדכן את כל המערכים שעוברים שורה.

על מנת לעדכן את מערך 2 עלינו להוריד אחד מהספירה של דרגת האפור של הפיקסל העליון הישן ולהוסיף אחד לספירה של דרגת האפור של הפיקסל התחתון החדש. מכיוון שאנו עושים זאת עבור כל טור וכל פעם שאנו עוברים שורה סה"כ זמן הריצה של כל העדכונים הללו הוא:

O(R*C)

בנוסף עלינו לבנות מחדש את מערך 1 עבור כל מעבר שורה.
את זאת ניתן לעשות ע"י חיבור של radius+1 ההיסטוגרמות הראשונות (הטורים) של מערך 2. מכיוון שאנו עושים זאת עבור כל מעבר שורה סה"כ זמן הריצה של כל העדכונים הללו הוא:

O(R*radius*MAX_G)

סה"כ נקבל שזמן הריצה הוא:

O(C*MAX_G+C*radius) + O(R*C*MAX_G) + O(R*C*MAX_G) + O(R*C) +O(R*radius*MAX_G)

ובהנחה שהרדיוס קטן ממספר השורות ומספר הטורים:

O(R*C*MAX_G)

שימו לב שזמן ריצה זה בוודאות טוב יותר משיטת מציאת החציון אם MAX_G קטן משמעותית מהרדיוס בריבוע (מכיוון שמדובר בסדר גודל, אי אפשר להבטיח את זה לסתם קטן אבל בוודאות מתישהוא זה ישתלם) .

קיימות שיטות נוספות למציאת כל החציונים אבל אנו לא נדבר עליהם בתרגיל זה.

הנחיות כלליות נוספות:

- כתבו את התרגיל כולו בקובץ GrayvalueImage.hpp (עליכם להשלים ולשנות אותו) ובקובץ GrayvalueImageTest.cpp כתבו גוגל טסט של התרגיל (חובה).
- בתרגיל זה אין להוסיף פקודות include נוספות
- בתרגיל זה יש להשתמש במערכי vector ולא במערכים פרימיטיביים
- רצוי (חלק מדרישות הסגנון) לכתוב גם פונקציות עזר כשצריך (יש להימנע מקוד כפול). למי שכתב פונקציות עזר – רצוי לכתוב גם פונקציות עזר לבדיקות
- הקוד שלכם צריך להתקמפל בלינוקס אובונטו ע"י קובץ ה-Makefile המצורף
- בדיקת הקוד לפני ההגשה, גם על ידי קריאתו וגם על ידי כתיבת בדיקות אוטומטיות היא אחריותכם. חישבו על מקרי קצה, חלק מהציון ניתן על עמידה בבדיקות אוטומטיות.

הנחיות Makefile:

- ניתן לקמפל את הקוד ע"י פקודת make
- הפקודה make installonce תתקין את google test

- הפקודה make clean תמחק קבצי object שהקומפיילר יצר
- הפקודה make zipfile תייצר עבורכם קובץ zip מוכן להגשה. כל שעליכם לעשות הוא למלא את תעודת הזהות שלכם בראש ה-Makefile בשדה הבא: ID=123456 במקום 123456
- הפקודה make checkzipfile תבדוק את קובץ ה-zip והקוד שבתוכו (בדיקה שהקובץ מכיל את מה שצריך ומתקמפל עם תוכנית ההדגמה, עליכם לכתוב בדיקה מקיפה יותר)
- הערה למתקדמים: מי שמעוניין שהקוד שלו ירוץ מהר במיוחד צריך להוציא מהערה את השורה:

```
CFLAGS= -O -DNDEBUG
```

ולשים בהערה את השורה:

```
CXXFLAGS += -D_GLIBCXX_DEBUG
```

<u>הנחיות הגשה</u>

כזכור עליכם להגיש קובץ ששמו מספר ת.ז. שלכם כמו שהיא מופיעה באתר המודל נקודה zip. לדוגמא, אם מספר ת.ז. שלי הוא 12345678 אז שם הקובץ יהיה: 12345678.zip

הקובץ יכיל את הקבצים הבאים בלבד:

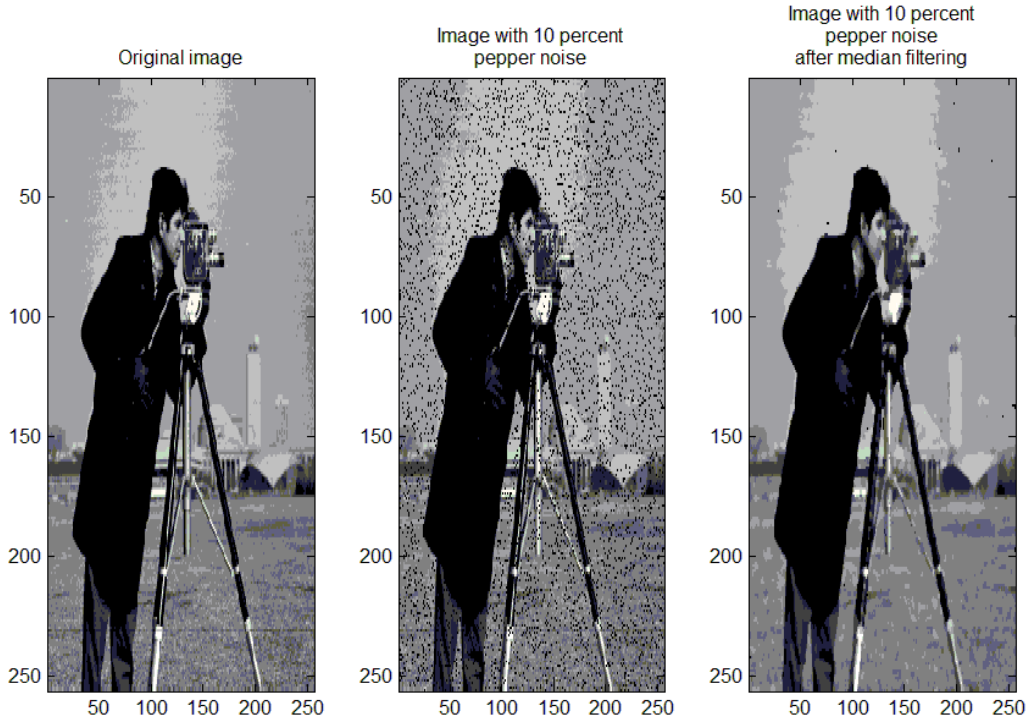
GrayvalueImage.hpp
GrayvalueImageTest.cpp

לאחר עדכון ה- ID בקובץ ה- Makefile פקודת make zipfile תיצור קובץ כזה. מומלץ ליצור את הקובץ כך על מנת למנוע טעויות.

<u>הערות העשרה (אין צורך לקרוא/להבין על מנת לקבל 100 בתרגיל)</u>

הפילטר בתרגיל זה יכול לנקות רעשי "מלח-פלפל". כלומר, שחלק מהפיקסלים הופכים לשחור לגמרי (0) או לבן לגמרי (MAX_G).
הקוד מכיל פונקציה המוסיפה רעש פלפל באחוז מסויים (פסאודו רנדומי) לפיקסלי התמונה. פונקציית ה main המצורפת קוראת תמונה מקובץ, מוסיפה לו רעש פלפל, כותבת את התמונה המתקבלת לקובץ חדש, מפעילה את ה median filter שעליכם לכתוב וכותבת את התמונה החדשה שהתקבלה לקובץ חדש (שלישי). מצורף גם קובץ Matlab אשר הרצה שלו תקרא את התוצאות שלכם ותציג אותם על המסך. התוצאה

שתתקבל תהיה דומה לתמונה המצורפת:



. על מנת להריץ את קובץ ה- matlab הריצו את matlab ואז כתבו:

cd name_of_folder

כאשר name_of_folder הוא שם התיקייה בה מופיע קובץ ה matlab וכל התוצאות. ואז פשוט רשמו show_results

שימו לב שהתוצאות לא יהיו זהות מכיוון שמדובר בתהליך פסאודו רנדומי.

אתם יכולים לשחק עם הרדיוס על מנת לקבל תוצאות שונות.

בהצלחה!!!