פרויקט מסכם תכנות מתקדם – תשע"ז סמסטר ב

תמונת רמות אפור בגודל $N \times M$ הינה מערך דו-מימדי מטיפוס unsigned char בגודל N שורות ו-M עמודות. כל תא במערך מכונה **פיקסל** ומכיל מספר בתחום 0-255 אשר מייצג את גוון האפור כאשר הערך 0 מייצג את הצבע השחור, הערך 255 מייצג את הצבע הלבן ויתר הערכים מייצגים גווני אפור בין שחור ללבן. סגמנט מוגדר כאוסף פיקסלים שכנים אשר השוני ברמות האפור של כל אחד מהם מרמת אפור נתונה X אינו עולה על סף נתון T. לדוגמא, האזור האפור בתמונה הבאה הוא סגמנט עבור X=95,T=10.

	0							M-1
0								
						100	110	
					85	95	40	
				80	100	94	50	
					70	**60**		
N-1								

על-מנת לייצג תמונה עושים שימוש בהגדרה הבאה:

```
typedef struct _grayImage {
  unsigned short        rows, cols;
  unsigned char         **pixels
} grayImage;
```

על-מנת לייצג מיקום בתמונה, עושים שימוש בהגדרה הבאה:

```
typedef unsigned short imgPos[2];
```

על-מנת לשמור סגמנט, עושים שימוש בעץ שנתון במבנה הבא:

```
typedef struct _treeNode{
   imgPos                position;
   treeNodeListCell      *next_possible_positions;  // רשימת מיקומים
} treeNode;

typedef  struct  _treeNodeListCell {
   treeNode                        * node;
   struct _treeNodeListCell        * next;
} treeNodeListCell;

typedef  struct  _segment {
   treeNode     * root;
} Segment;
```

<u>שימו לב:</u> על-מנת שהגדרות אלו יעברו קומפילציה יש להשתמש ב-forward declaration. כחלק מהפרויקט, עליכם למצוא באינטרנט את אופן השימוש בזה.

<u>סעיף 1</u>

כתבו את הפונקציה:

grayImage *readPGM(char *fname)

הפונקציה מקבלת שם של קובץ טקסט אשר מכיל תמונה בפורמט PGM , מייצרת תמונה מטיפוס **grayImage** ומחזירה מצביע אליו.

מבנה תמונה בפורמט PGM הוא כדלקמן: בשורה הראשונה מופיעה המילה P2. בשורה השניה מופיע מספר השורות ואחריו מספר העמודות. בשורה השלישית מופיע ערך רמת האפור המקסימלי. בשורה הרביעית ואילך מופיעים ערכי האפור של הפיקסלים שורה אחר שורה.

להלן דוגמא לתמונה בגודל 10 שורות ו20 עמודות עם 3 רמות אפור (0 שחור, 1 אפור, 2 לבן)

```
P2
20  10
2
1  1  1  1  0  0  0  0  2  2  2  2  1  1  1  1  0  0  0  0
1  1  1  1  0  0  0  0  2  2  2  2  1  1  1  1  0  0  0  0
1  1  1  1  0  0  0  0  2  2  2  2  1  1  1  1  0  0  0  0
1  1  1  1  0  0  0  0  2  2  2  2  1  1  1  1  0  0  0  0
1  1  1  1  0  0  0  0  2  2  2  2  1  1  1  1  0  0  0  0
2  2  2  2  1  1  1  1  0  0  0  0  2  2  2  2  1  1  1  1
2  2  2  2  1  1  1  1  0  0  0  0  2  2  2  2  1  1  1  1
2  2  2  2  1  1  1  1  0  0  0  0  2  2  2  2  1  1  1  1
2  2  2  2  1  1  1  1  0  0  0  0  2  2  2  2  1  1  1  1
2  2  2  2  1  1  1  1  0  0  0  0  2  2  2  2  1  1  1  1
```

<u>סעיף 2</u>

בסעיף זה יש לכתוב את הפונקציה:

Segment findSingleSegment(grayImage *img, imgPos start,
unsigned char threshold)

הפונקציה מוצאת סגמנט ע"י סריקת התמונה החל מהמיקום start. הפונקציה תבדוק את ערכי האפור של כל 8 השכנים של start (ימין, שמאל, מעלה, מטה, ובאלכסונים). לפיקסלים אשר בשורה הראשונה והאחרונה (כך גם עבור העמודות) יש פחות מ-8 שכנים. אלו אשר הפרש רמת האפור ביניהם לזו של start קטן מ-threshold יוכנסו כצמתים לעץ. התהליך ימשיך בצורה רקורסיבית מהשכנים. בשורש העץ יהיה המיקום start.

<u>שימו לב:</u> אסור שבעץ יופיע מיקום מסוים יותר מפעם אחת כלומר כאשר בודקים את השכנים יש לבדוק בצורה יעילה ככל שניתן האם הם טרם צורפו לסגמנט ורק אז לבדוק את רמות האפור שלהם.

סעיף 3

נתונה ההגדרה הבאה לאחסון מיקומים ברשימה מקושרת:

```
typedef struct _imgPosCell {
   imgPos                position;
   struct _imgPosCell *next, *prev;
} imgPosCell;

typedef struct _imgPosList {
         imgPosCell    *head, *tail;
} imgPosList;
```

בסעיף זה יש לכתוב את הפונקציה:

int findAllSegments(grayImage *img, unsigned char threshold, imgPosList **segments)

הפונקציה מוצאת את כל הסגמנטים בתמונה ומחזירה אותם בארגומנט segments. הערך המוחזר של הפונקציה יהיה גודל המערך. על-מנת למצוא את כל הסגמנטים יש להתחיל ממיקום כלשהוא בתמונה ולמצוא את הסגמנט המתחיל ממנו. אח"כ יש לבחור מיקום מאלה שאינם שייכים לסגמנט שכבר מצאנו ולמצוא את הסגמנט שמתחיל ממנו. כך יש להמשיך עד אשר כל פיקסל בתמונה שייך לסגמנט מסוים.

סעיף 4

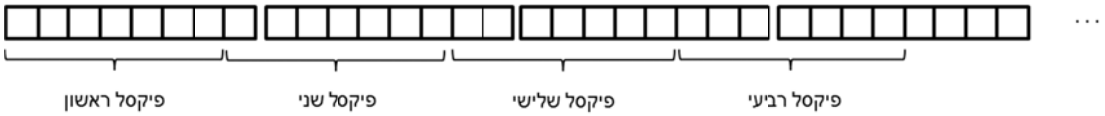
בסעיף זה יש לייצר תמונה שבה כל הפיקסלים בכל סגמנט יכילו את אותה רמת אפור. את רמת האפור של כל סגמנט יש לקבוע בהתאם לכמות הסגמנטים size באופן הבא: ערך האפור של הסגמנט ה-i (מספרי הסגמנטים מתחילים מאפס) יהיה $\left\lfloor i * \frac{255}{size-1} \right\rfloor$. לדוגמא, אם יש שלושה סגמנטים, ערך האפור של הראשון יהיה 0, של השני יהיה 127 ושל השלישי 255. הפונקציה תקבל רשימה של סגמנטים כמו זו שנוצרת בסעיף 2.

grayImage *colorSegments(imgPosList *segments, unsigned int size)

סעיף 5

הוחלט לשמור באופן חסכוני תמונת רמות אפור. לשם כך קובעים ערך אפור מקסימלי $new\_max < 128$ ומשנים את ערך של פיקסל p כך שיהיה בטווח החדש $[0, new\_\max]$ לפי הנוסחה הבאה $p_{new} = \left\lfloor \frac{p \times new\_max}{255} \right\rfloor$. לדוגמא, עבור $new\_max = 127$ פיקסל שערכו 255 ישונה ל-127, פיקסל שערכו 192 ישונה ל-95 וכן הלאה. אין לשנות את התמונה שמתקבלת כפרמטר.

יש לשמור את התמונה עם טווח רמות האפור המוקטן בקובץ בינארי באופן הבא: בתחילת הקובץ יהיו שני משתנים מסוג unsigned short אשר יכילו את מספר השורות והעמודות של התמונה. אח"כ יהיה בית אשר יכיל את רמת האפור המקסימלית. ואח"כ יישמרו הפיקסלים ברצף שורה אחר שורה כאשר כל פיקסל מיוצג ע"י מספר הביטים המינימלי הנדרש. אם $new\_max = 127$, כל פיקסל יישמר בעזרת שבעה ביטים באופן הבא:



יש לכתוב את הפונקציה:

```
void saveCompressed( char *file_name, grayImage *image,
                     unsigned char max_gray_level)
```

הפונקציה מקבלת שם קובץ דחוס, תמונה image וערך אפור מקסימלי max_gray_level.

סעיף 6

בסעיף זה יש לכתוב את הפונקציה:

```
void convertCompressedImageToPGM(
        char *compressed_file_name, char *pgm_file_name)
```

הפונקציה מקבלת שם של קובץ בינארי עם תמונה דחוסה כפי שיוצרים בסעיף 5 ומייצרת קובץ PGM.

סעיף 7

בסעיף זה עליכם ליצור תפריט אשר יאפשר את הפעלת הסעיפים שכתבתם.

1. Read an image file in PGM format
2. Find all segments
3. Color the segments
4. Save the colored image in a compressed format
5. Compress and save the original image in a compressed format
6. Convert a compressed image to PGM format

באופציה 1 יש לקלוט מהמשתמש שם קובץ תמונה בפורמט PGM, ולטעון אותו לזיכרון.
באופציה 2 יש למצוא את כל הסגמנטים בתמונה שקראתם בסעיף 1. אם טרם נטענה תמונה יש להודיע למשתמש ולטעון אותה.
באופציה 3 יש לייצר תמונה שבה כל הפיקסלים בכל סגמנט יכילו את אותה רמת אפור. אם טרם נמצאו הסגמנטים, יד למצוא אותם.
באופציה 4 יש לשמור את התמונה שנוצרה באופציה 3 בקובץ דחוס.
באופציה 5 יש לקלוט מהמשתמש שם קובץ וערך אפור מקסימלי, לדחוס ולשמור את התמונה בקובץ.
באופציה 6 יש לקלוט מהמשתמש שם קובץ דחוס ושם קובץ נוסף ולהמיר את הקובץ הדחוס לקובץ בפורמט PGM.

הערה:

על-מנת לראות כיצד תמונת PGM נראית, תוכלו לעשות שימוש באחת מהתוכנות החינמיות לדוגמא irfanview.