אוניברסיטת תל אביב
הפקולטה להנדסה
תכנות 2 בשפת C – 0512.1820
סמסטר ב' תש"פ

מבחן סופי – מועד א'

Final Exam - Moed A

- יש לענות על כל השאלות וכל הסעיפים.
- משך הבחינה 3 שעות.
- אסור שימוש בחומר עזר ואסור שימוש במחשבון.
- נא לענות בטופס הבחינה. המחברות הן לטיוטה ולא תיבדקנה.

- Answer all questions and sections.
- Exam length 3 hours.
- No usage of aid material, nor a calculator.
- Please answer in the exam form. Notebooks will not be checked.

בהצלחה,

פרופ' יובל שביט, ד"ר רני הוד, ד"ר לזר מילנקוביץ'

Good luck,

Prof Yuval Shavitt, Dr. Rani Hod, Dr. Lazar Milenkovic

שאלה Question 1

מערכת תצוגה מציגה תמונה בגודל PIC_SZ*PIC_SZ פיקסלים (פיקסל הוא 'נקודה' בודדת במסך). כל פיקסל מיוצג ע"י 3 בתים, עבור שלושת הצבעים אדום (R), ירוק (G) וכחול (B). הניחו שהתמונה נשמרת במערך דו-מימדי מטיפוס unsigned char בגודל של [PIC_SZ][3*PIC_SZ], המבנה של שורה במערך בציור למטה, כאשר R1, G1 ו- B1 הם הנתונים של פיקסל 1; R2, G2 ו- B2 הנתונים של פיקסל 2 וכן הלאה.

A display system displays square pictures of size PIC_SZ*PIC_SZ pixels (A pixel is a single 'dot' in the screen). Each pixel is represented by **3 bytes**, for the three colors Red (R), Green (G), and Blue (B). We assume that the picture is kept in an bi-dimensional unsigned char array of size [PIC_SZ][3*PIC_SZ], the structure of an array row is depicted below, where R1, G1, B1 are the data for the pixel 1; R2, G2, B2 the data for pixel 2; etc.

R1	G1	B1	R2	G2	B2	R3	...

1. אנו מעוניינים לדחוס את הייצוג כך שכל פיקסל ייוצג ע"י בית בודד, כך ששלוש הסיביות המשמעותיות ביותר יילקחו משלוש הסיביות המשמעותיות ביותר מהבית האדום, שתי הסיביות הבאות יהיו שתי הסיביות המשמעותיות ביותר של הבית הירוק, ושלוש הסיביות הפחות משמעותיות תלקחנה משלוש הסיביות המשמעותיות ביותר מהבית הכחול.

A. We want to compress this presentation such that each pixel will be represented by a single byte, such that the MS three bits will be the 3 MS bits from the R bytes, the following 2 bits will be the 2 MS bits of the G byte and the 3 LS bits will be the 3 MS bits of the B byte.

כיתבו פונקציה המקבלת מצביע למערך של שלושה בתים (RGB) ומחזירה unsigned char בייצוג החדש.

Write a function that accepts a pointer to an array of 3 bytes (RGB) and returns an unsigned char in the new representation

```
#define PIC_SZ 30

unsigned char pix_convert(unsigned char *rgb) {
	unsigned char ret = 0;
	ret |= (rgb[0] & 0xe0);
	ret |= (rgb[1] & 0xc0) >> 3;
	ret |= (rgb[2] & 0xe0) >> 5;

	return ret;

}
```

ב. כיתבו פונקציה המקבלת מצביע לשורה אחת של תמונה ומחזירה מצביע לשורה דחוסה (שאותה יש להקצות דינמית).

B. Write a function that accepts a pointer to a single line of a picture and returns a pointer to a compressed line (that should be dynamically allocated)

```
unsigned char *mk_line(unsigned char *line) {
	int i = 0;
	unsigned char * ret = malloc(sizeof(char)*PIC_SZ);
	if (!ret) {
		printf("Memory allocation failed\n");
		return NULL;
	}
	for (i = 0; i < PIC_SZ; i++)
		ret[i] = pix_convert(line + 3 * i);
	return ret;
}
```

ג. כיתבו פונקציה המקבלת מצביע למערך של תמונה דו-מימדית ומחזירה מצביע למערך דו-מימדי דחוס שהוקצה באופן דינמי. רמז: השתמשו במערך מצביעים חד-מימדי כשכל איבר מצביע למערך המייצג שורה בתמונה הדחוסה.

C. Write a function that accepts a pointer to a bi-dimensional picture array and return a pointer to a compressed array that was dynamically allocated. Hint: use a vector of pointers, each pointing to an array that represents a single line.

```
unsigned char **mk_pic(unsigned char pic[][3*PIC_SZ]) {
    unsigned char **ret = malloc(PIC_SZ * sizeof(unsigned char*));
    if (!ret) {
        printf("Memory allocation failed\n");
        return NULL;
    }
    int i;
    for (i = 0; i < PIC_SZ; i++) {
        ret[i] = mk_line(pic[i]);
    }
    return ret;
}
```

Question 2 שאלה

A. Implement the function `strrchr` (which exists in the standard C library, BTW).

Arguments: a string `s` and a character `c`.
Return value: if `s` does not contain `c` (or `s` is `NULL`), return `NULL`;
Otherwise return a pointer to the **last** occurrence of `c` in `s`.

1. ממשו את הפונקציה `strrchr` (שקיימת בספריה הסטנדרטית, אגב).

ארגומנטים: מחרוזת `s` ותו `c`.
ערך חזרה: אם התו `c` לא נמצא במחרוזת `s` (או ש-`s` עצמו `NULL`) מוחזר `NULL`;
אחרת מוחזר מצביע למופע **האחרון** של `c` ב-`s`.

```
char *strrchr(char *s, char c)
{
	if (!s)
		return NULL;
	char *result = NULL;
	while (*s) {
		if (*s == c)
			result = s;
		++s;
	}
	return result;
}
```

B. Implement a function named `filter_file`.

The purpose of the function is to read text lines from one file and write them to another file, with the following changes (test for the first condition that holds, in this order):

1. If the character `$` is in the line, do not write the line to the output file.
2. If the character `#` is in the line, write only the part after the last occurrence of `#` (not including `#` itself).
3. If the character `"` appears at least twice in the line, write only the part between the first and the last occurrences of `"` (not including both `"`).
4. Otherwise, write the line as-is to the output file.

<u>Arguments</u>: two strings containing the file names.
<u>Return value</u>: zero on success; nonzero on error.

The function fails if it cannot open either the input or output file.
You may assume each text line contains at most 199 characters (including the terminal `'\n'`).
You may assume read/write operations always succeed (if the file was opened successfully).
You may of course use `strrchr` from part A, and also the standard function `strchr`, which does the same as `strrchr` except it returns a pointer to the **first** occurrence of `c`.

2. ממשו פונקציה בשם `filter_file`.

מטרת הפונקציה לקרוא שורות טקסט מקובץ אחד ולכתוב אותן לקובץ אחר, עם השינויים הבאים (יש לבדוק לפי הסדר מה התנאי הראשון שמתקיים):

1. כאשר שורה מכילה את התו `$`, לא כותבים אותה לקובץ הפלט.
2. כאשר שורה מכילה את התו `#`, כותבים רק את החלק שאחרי המופע האחרון של `#` (לא כולל `#` עצמו).
3. כאשר שורה מכילה את התו `"` לפחות פעמיים, השורה שנכתבת לקובץ הפלט תכיל רק את החלק שבין ה-`"` הראשון ועד ה-`"` האחרון (לא כולל שני ה-`"`).
4. אחרת השורה נכתבת כמו שהיא לקובץ הפלט.

<u>ארגומנטים</u>: שתי מחרוזות המציינות שמות של קבצים.
<u>ערך חזרה</u>: אפס אם הפונקציה הצליחה; nonzero אם הפונקציה נכשלה.

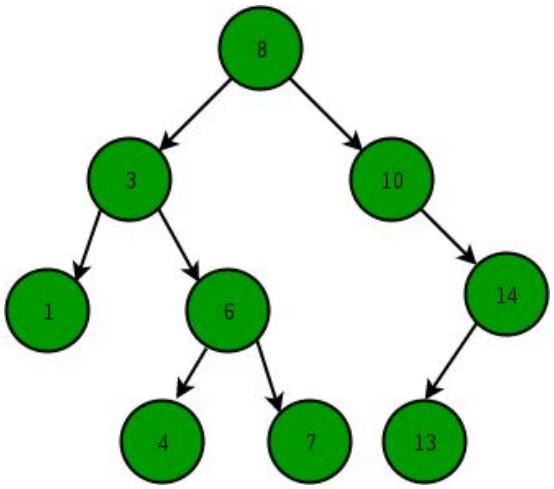
הפונקציה נכשלת אם יש בעיה בפתיחת קובץ הקלט או הפלט.
ניתן להניח שכל שורת טקסט בקובץ הקלט היא באורך של 199 תוים לכל היותר (כולל ה-`'\n'` בסופה).
ניתן להניח שפעולות קריאה/כתיבה של קובץ מצליחות (בהנחה שלא הייתה בעיה בפתיחתו).
מותר כמובן להשתמש ב-`strrchr` מסעיף א', וגם בפונקציה `strchr` שעושה אותו דבר כמו `strrchr` אבל מחזירה מצביע למופע **הראשון** של התו `c`.

```
#define MAX_LINE 200
int filter_file(const char infile[], const char outfile[])
{
	FILE *fi=NULL, *fout=NULL;
	fin = fopen(infile, “r”);
	if (fin == NULL)
		return -1; // cannot open input file
	fout = fopen(outfile, “w”);
	if (fout == NULL) {
		fclose(fin);
		return -2; // cannot open output file
	}
	// copy loop
	char line[MAX_LINE];
	while (!feof(fin)){
		fgets(line, MAX_LINE, fin);
		char *a, *b;
		// case 1
		if (strchr(line, ‘$’))
			continue; // write nothing
		// case 2
		if (a = strrchr(line, ‘#’)) {
			fputs(a+1, fout);
			continue;
		}
		// case 3
		a = strchr(line, ‘”’);
		b = strrchr(a, ‘”’);
		if (a && b != a) {
			b[0] = ‘\n’;
			b[1] = ‘\0’;
			fputs(a+1, fout);
			continue;
		}
		// case 4
		fputs(line, fout);
	}
	fclose(fin);
	fclose(fout);
	return 0; // all OK
}
```

שאלה Question 3

עץ בינארי הוא מבנה נתונים שימושי המשמש לבניה יעילה של מילונים וקבוצות. דוגמא לעץ בינארי מצוירת מטה.

A binary tree is a useful data structure when it comes to the efficient construction of dictionaries and sets. An example binary tree is shown in the figure below.



בדומה לרשימות מקושרות, ניתן לייצג עץ ע"י שימוש במבנים ומצביעים. ההבדל העיקרי במעבר מרשימה מקושרת לעץ הוא שצומת בעץ מכיל שני מצביעים: ימני ושמאלי. הקוד למטה מגדיר מבנה לעץ חיפוש בינארי.
Similarly to the linked lists, we can represent a tree using structs and pointers. The key difference is that each tree node contains two pointers, left and right. The following code represents the struct definition for a binary search tree node.

```
typedef struct tree_node {
    int data;
    struct BSTNode *left, *right;
} BSTNode;
```

בציור למעלה, לצמת 8 יש מצביע שמאלי המצביע לצמת 3 ומצביע ימני לצומת 10. צמת 8 נקרא השורש של העץ מכיון שניתן להגיע לכל הצמתים בעץ ע"י מהלך המתחיל מצומת זה. השורש מקביל קונספטואלית לראש הרשימה המקושרת. הצמתים בתחתית, 1, 4, 7 ו- 13, נקראים עלים מכיון שגם המצביע הימני שלהם וגם המצביע השמאלי הם NULL. שימו לב שקיימים צמתים שמצביע אחד שלהם הוא NULL והאחר מצביע לצומת קיים בעץ.
In the image above, a node with value 8 has left pointer pointing to the node with value 3, and right pointer to the node with value 10. Node 8 is called the **root** of the tree since it is possible to reach all the elements in a traversal from it. The concept of a special element is similar to the head of linked list. Elements at the bottom, with values 1, 4, 7, and 13 are called **leaves** of the binary search tree and they have both left and right pointers pointing to `NULL`. Finally, notice that some nodes can have one pointer pointing to another node and the other pointer set to `NULL`. Example in the image above is node 14, which has its left pointer pointing to the node with value 13 and the right pointer set to `NULL`.

1. כיתבו פונקציה שעוברת על כל צמתי העץ ומדפיסה את המספרים בשדה data . הפונקציה מקבלת כפרמטר מצביע לשורש העץ. רמז: השתמשו ברקורסיה.

A. Write a function that traverses all the elements in the tree and prints their data fields. The function takes an argument that is a pointer to the root of the tree.

Hint: use recursion!

```
void traverse(BSTNode *root) {
    if (root == NULL) {
        return;
    }
    traverse(root->left);
    printf("%d\n", root->data);
    traverse(root->right);
}
```

2. כיתבו פונקציה המקבלת מצביע לשורש העץ וערך שלם ומחזירה מצביע לצומת המכיל את הערך, או NULL אם הערך לא קיים. הניחו שהעץ ממויין, משמע ערכי כל הצמתים משמאל לצומת קטנים מערכו, וכל הצמתים מימיו לו גדולים מערכו.

Using the ideas similar to the previous exercise, write a function that receives an integer `val`, and a pointer to a root node and returns a pointer to an element that has data field equal to `val`, or `NULL` if such an element does not exist in the tree. Assume that the tree is sorted, namely all the nodes to the left of a node are smaller that it and all the nodes to the right are larger..

```
BSTNode  *find(BSTNode *root, int val) {
    if (root == NULL) {
        return NULL;
    }
    if (root->data == val) {
        return root;
    }
    BSTNode *temp = find(root->left, val);
    if (temp != NULL) {
        return temp;
    }
    return find(root->right, val);
}
```

ג. כיתבו פונקציה המקבלת מצביע לצומת של עץ, ומוסיפה לו צאצא ימני הגדול ממנו ב-1.

C. Write a function that given a pointer to tree leaf, adds a new right child with a value larger by 1.

```
void insert(BSTNode *tree_node) {
      BSTNode * new_node = NULL;
      new_node = malloc(sizeof(BSTNode));

      if (!new_node)
      {
            printf("Memory allocation failed\n");
            return NULL;
      }

      new_node->right = NULL;
      new_node->left = NULL;
      new_node->data = tree_node->data + 1;

      tree_node->right = new_node;

}
```