ת.ז._____
מס. מחברת _____

מבחן דוגמה
סמסטר ב' תשע"ו

משך הבחינה: שעתיים

בחינה בקורס: פרויקט תוכנה **03682161**

מרצים: פרופ' דניאל כהן-אור ופרופ' דן הלפרין

הנחיות כלליות לבחינה:

- בראש כל עמוד של טופס המבחן יש לציין את מספר תעודת הזהות.
- בבחינה ארבע שאלות (פתוחות ואמריקאיות) בעלות ניקוד משתנה, בסך של 100 נק'.
- את התשובות לשאלות 1-2 יש למלא **במקום המסומן בלבד.**
- את התשובות לשאלות 3-4 יש למלא **בטבלה המיועדת לכך (בעמוד זה)**
- **אם סומנו שתי עמודות התשובה לא תתקבל.**

בהצלחה !

טבלת תשובות לשאלות 3-4

	א'	ב'	ג'	ד'
שאלה 3				
שאלה 4				

נימוק קצר עבור שאלות 3-4

שאלה 3: _____

שאלה 4: _____

שאלה 1 (25 נק')

בשאלה זו נעבד קובץ טקסט מיוחד אשר מכיל שמות של לקוחות ביחד עם סכום הקניות בשקלים של כל לקוח. מבנה הקובץ הוא כדלקמן:

John Snow#1245
Jaime Lannister#1245
Eddard Stark#19
Tyrion Lannister#17288

כלומר כל שורה בקובץ הינה מהצורה הבאה: <Client Name>#<Net Buy>

להלן מימוש חלקי של תוכנית אשר מקבלת קובץ מכירות (command line argument כ) ומדפיסה את שמות הלקוחות ביחד עם סכום הקניות שלהם ממוינים בסדר יורד לפי ס"כ הקניות (אם לשני לקוחות אותו סכום יש להדפיס בסדר ליקסוגרפי עולה). ראו דוגמה למטה:

1 : Tyrion Lannister : 17288
2 : Jaime Lannister : 1245
3 : John Snow : 1245
4 : Eddard Stark : 19

הנחות:

- הניחו כי הקובץ תקין והמשתמש אכן מכניס שם של קובץ תקין ברשימת הארגומנטים

המשך בעמוד הבא

```c
#include <string.h>
#include <stdio.h>
#include <stdlib.h>

#define LINE_LENGTH 1024

typedef struct client_t {
     char* name;
     int netBuy;
} Client;

Client* parseLine(char* line) {
     int i = 0, lineLen = strlen(line), val;
     Client* res = NULL;
     for (; i < lineLen; i++) {
          if (line[i] == '#') {
               line[i] = _________;
               val = atoi(________________);
          }
     }
     res = (Client*) malloc(sizeof(*res));
     res->name = (char*) malloc(sizeof(char) *
                                          (___________));
     strcpy(____________, ______________);
     res->netBuy = ___________________;
     return res;
}
```

```
int compareClients(const void* a, const void* b) {
    ____________________________________________________
    ____________________________________________________
    ____________________________________________________
    ____________________________________________________
    ____________________________________________________
    ____________________________________________________
    ____________________________________________________
    ____________________________________________________
    ____________________________________________________
    ____________________________________________________
    ____________________________________________________
    ____________________________________________________
}
```

```
Client** getClients(FILE* fp, int* size) {
    char line[LINE_LENGTH];
    int numOfLines = 0;
    Client** res = NULL;
    while (fgets(line, LINE_LENGTH, fp)) {
        numOfLines++;
        res = (Client**) realloc(_____________,
                                 _________________________);
        res[_________________] = parseLine(_____________);
    }
    *size = numOfLines;
    return res;
}

int main(int argc, char* argv[]) {
    FILE* fp = fopen(argv[1], "r");
    int numOfClients = 0;
    Client** clients = NULL;
    int i = 0;
    clients = getClients(_____________, _________________);
    qsort(_______________________________________________,
          _______________________________________________,
          _______________________________________________,
          _______________________________________________);
    for (; i < numOfClients; i++) {
        printf("%d : %s : %d\n", i + 1,
                                 clients[i]->name,
                                 clients[i]->netBuy);
        free(clients[i]->name);
        free(clients[i]);
    }
    free(clients);
    fclose(fp);
    return 0;
}
```

שאלה 2 (15 נק')

בשאלה זו נכתוב פונקציה אשר ממינת רשימה מקושרת דו-כיוונית.

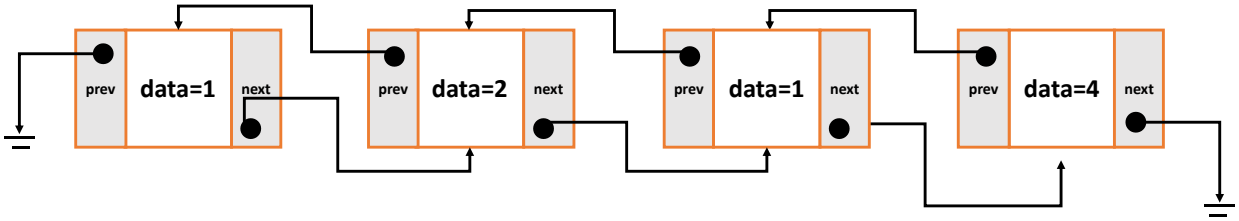
להלן הגדרה של מבנה אשר מייצג רשימה דו-כיוונית של ערכים שלמים (INTEGERS):

```
typedef struct node_t {
     int data;
     struct node_t* next;
     struct node_t* prev;
} ListNode;
```

בנוסף, הניחו כי:

- השדה **prev** של האיבר שבראש הרשימה הינו **NULL**
- השדה **next** של האיבר שבסוף הרשימה הינו **NULL**
- אם איבר הוא גם בראש הרשימה וגם בסוף הרשימה (כלומר מייצג רשימה של איבר אחד) אזי שני השדות **prev** ו **next** הינם **NULL**
- רשימה ריקה מיוצגת על ידי הערך **NULL**
- הניחו כי הרשימה נבנת באופן דינמי, כלומר האיברים נוצרו על ידי קריאה לפונקצית malloc

ראו למטה דיאגרמה אשר מייצגת רשימה של 4 איברים:



בדוגמה למעלה, האיבר הראשון (ראש הרשימה) מכיל את הערך 1, האיבר השני מכיל את הערך 2 והאיבר השלישי מכיל את הערך 1 והאיבר הרביעי (סוף הרשימה) מכיל את הערך 4.

סעיף א (15 נק')

כתבו פונקציה ריקורסיבית אשר מקבלת רשימה ומחזירה את גודלה. הניחו כי הגודל של רשימה ריקה הוא 0.

```
int listSize(ListNode* list) {
    ____________________________________________________
    ____________________________________________________
    ____________________________________________________
    ____________________________________________________
    ____________________________________________________
    ____________________________________________________
}
```

סעיף ב (15 נק')

השלימו את הפונקציה הריקורסיבית הבאה אשר מקבלת שתי רשימות ממוינות וממזגת אותן לרשימה מקושרת ממוינת. בנוסף הפונקציה מקבלת prev שתחילה מאותחל ל NULL (השתמשו בארגומנט זה כדי לציין את שדה ה PREV של ראש הרשימה הממוזגת בקריאה הריקורסיבית הנוכחית). על הפונקציה להחזיר את ראש הרשימה הממוזגת החדשה. (הניחו כי prev=NULL בקריאה הראשונה).

```
ListNode* mergeSortedLists(ListNode* list1,
                           ListNode* list2,ListNode* prev) {
    ListNode* head = NULL;
    if (!list1) {
        if (list2!=NULL){
            list2->prev = prev;
        }
        return list2;
    }
    if (!list2) {
        if (list1!=NULL){
            list1->prev = prev;
        }
        return list1;
    }
    if (list1->data < list2->data) {
        ____________________________________________________
        ____________________________________________________
        ____________________________________________________
    } else {
        ____________________________________________________
        ____________________________________________________
        ____________________________________________________
    }
    return head;
}
```

סעיף ג (15 נק')

השלימו את הפונקציה הריקורסיבית הבאה (זהו מימוש MERGE SORT עבור רשימות מקושרות) אשר מקבלת רשימה מקושרת וממיינת אותה. הפונקציה מחזירה את ראש הרשימה הממויונת.

```
ListNode* sortList(ListNode* list) {
    int count = 0;
    ListNode* leftList, *rightList, *current;
    if (!list) {
        return NULL;
    }
    count = listSize(list) / 2;
    if (count == 0) {
        ____________________________________________
    }
    ____________________________________________
    ____________________________________________
    ____________________________________________
    ____________________________________________
    ____________________________________________
    ____________________________________________
    ____________________________________________
    ____________________________________________
    ____________________________________________
    ____________________________________________
    ____________________________________________
    ____________________________________________
}
```

שאלה 3 (15 נק')

איזו טענה מהטענות הבאות נכונה עבור קטע הקוד הבא כאשר שורת הקימפול של התוכנית הינה:

```
>> gcc -std=c99 -Wall -Werror main.c
```

```c
#include <stdio.h>
#include <stdlib.h>

void foo(char* str) {
	char* temp = NULL;
	if (str[0] == 'a') {
		free(NULL);
	} else if (str[0] == 'b') {
		while (1) {
		}
	} else if (str[0] == 'c') {
		putchar(*temp);
	} else {
		putchar(str[3]);
	}
}

int main() {
	int arr[4] = { 'a', 'b', 'c', 'd' };
	foo((char*)arr+1);
	return 0;
}
```

א. התוכנית לא מתקמפלת עקב המרה לא חוקית

ב. התוכנית קורסת עקב גישה לא חוקית לזיכרון

ג. התוכנית נכנסת ללולאה אינסופית

ד. התוכנית מדפיסה את התו 'b'

שאלה 4 (15 נק')

איזו טענה מהטענות הבאות נכונה עבור קטע הקוד הבא כאשר שורת הקימפול של התוכנית הינה:

```
>> gcc -std=c99 -Wall -Werror main.c
```

```c
#include <stdio.h>
#include <stdlib.h>

void foo(char* str) {
	char* temp = NULL;
	if (str[0] == 'a') {
		while(1){}
	} else if (str[0] == 'b') {
		free(NULL);
	} else if (str[0] == 'c') {
		putchar(*temp);
	} else {
		putchar(str[3]);
	}
}

int main() {
	int a = (((((('a'<<8)|'b')<<8)|'c')<<8)|'d';
	foo((char*)&a + 1);
	return 0;
}
```

א. התוכנית לא מתקמפלת עקב המרה לא חוקית

ב. התוכנית קורסת עקב גישה לא חוקית לזיכרון

ג. התוכנית נכנסת ללולאה אינסופית

ד. התוכנית מדפיסה את התו 'b'