Question 1 (36 points): שאלה 1 (36 נקודות):

► A file named `lego.txt` contains data about select LEGO store locations in California, Colorado, and New York (source: https://www.lego.com/en-us/stores/directory). Each line contains information for a different store location.

◄ קובץ בשם `lego.txt` מכיל מידע לגבי מבחר חנויות לגו בקליפורניה, קולורדו וניו-יורק (מקור: https://www.lego.com/en-us/stores/directory). כל שורה מכילה מידע עבור חנות שונה.

The line format for the file `lego.txt` פורמט של שורה בקובץ

<zip><comma>**<name>**<colon>**<features>**

- **<zip>** specifies the zip code of the location. The zip code consists of exactly 5 digits.
- **<name>** is the name of the store location. The name consists of letters (upper or lower case) and space characters (`' '`).
- **<features>** is an optional list of store features. If the list is empty, then nothing follows the colon character. Each feature in the list consists of letters (upper or lower case) and space characters (`' '`). If the list of features contains more than one feature, then consecutive features are separated using a single comma character (`','`).
- <comma> and <colon> are the comma `','` and colon `':'` characters.
- You may assume that all lines in the file are written in this format and that there are no blank lines in the file.

- **<zip>** מציין את המיקוד של החנות. המיקוד מורכב מחמש ספרות בדיוק.
- **<name>** הוא שם החנות. שם החנות מורכב מאותיות (גדולות או קטנות) ותווי רווח (`' '`).
- **<features>** היא רשימה אפשרית של תכונות לחנות. אם הרשימה ריקה, אז לא מופיע דבר אחרי תו הנקודתיים. כל תכונה ברשימה מורכבת מאותיות (גדולות או קטנות) ותווי רווח (`' '`). אם רשימת התכונות כוללת יותר מתכונה אחת, אז תכונות עוקבות מופרדות על ידי תו פסיק בודד (`','`).
- <comma> ו <colon> הם תווי הפסיק `','` והנקודתיים `':'`.
- ניתן להניח שכל השורות ב `lego.txt` כתובות בפורמט הזה ושאין שורות ריקות בקובץ.

► Example contents of the file: `lego.txt` :תוכן קובץ לדוגמא ◄

```
10020,Fifth Avenue:Brick Lab,Mosaic Maker,Minifigure Factory
12203,Crossgates Mall:
92802,Downtown Disney District:Mosaic Maker,Minifigure Factory
13204,Destiny USA:Curbside Pickup
13204,Eastview Mall:Curbside Pickup
10010,Flatiron District:Mosaic Maker
95678,Westfield Galleria At Roseville:
94103,Westfield San Francisco Center:Curbside Pickup
91303,Westfield Topanga:Curbside Pickup
92122,UTC Mall:Mosaic Maker
80401,Colorado Mills:
80124,Park Meadows:
94588,Stoneridge Mall:Curbside Pickup
14225,Walden Galleria:Curbside Pickup
```

► For each of the piped sequences of commands specified in (a)-(c) below, write its expected output given the example file above. You may provide a brief explanation for your answer to describe your rationale.

◄ לכל אחד מרצפי הפקודות המשורשרות המצוינים בסעיפים א'-ג' למטה, כתבו את הפלט הצפוי שלו בהינתן הקובץ לדוגמא למעלה. אתם רשאים להוסיף הסבר קצר לתשובה שלכם.

a) (6 pts.) **א) (6 נק')**

```
head -n2 lego.txt | cut -d"," -f1 | \
      tr -d "\n" | tr -s [[:digit:]]
```

Expected output: פלט צפוי:

b) (6 pts.) **ב) (6 נק')**

```
cut -d"," -f2 lego.txt | cut -d":" -f1 | \
   paste - lego.txt -d"_"  | cut -d"," -f1 | tail -n2
```

Expected output: פלט צפוי:

c) (6 pts.) **ג) (6 נק')**

```
grep ":$" lego.txt | tail -n2 | \
   sed -r 's/,(C.*)[[:space:]].*/_\1_/'
```

Expected output: פלט צפוי:

► Solve the tasks in (d)-(f) as instructed. Your code should work on every input file `lego.txt` that satisfies the specification on page 1, and not just the example file on page 2. Use only commands and options that were covered in the lectures and/or tutorials.

◄ פתרו את המשימות בסעיפים ד'-ו' על פי ההוראות. הקוד שלכם אמור לעבוד על כל קובץ קלט `lego.txt` העומד בתיאור בעמוד 1 ולא רק על הקובץ לדוגמא בעמוד 2. השתמשו רק בפקודות ואופציות שנלמדו בהרצאות ו/או בתרגולים.

d) (6 pts.) Write a piped sequence of commands that prints the name of the store location with the largest zip code as a five-digit number. When applied to the example file from page 2, your piped sequence of commands should print the following line of text:

```
Westfield Galleria At Roseville
```

ד) (6 נק') כתבו רצף פקודות משורשרות שמדפיס את השם של החנות עם המיקוד הכי גדול כמספר בן חמש ספרות. כאשר הוא מופעל על הקובץ לדוגמא מעמוד 2, רצף הפקודות המשורשרות שלכם אמור להדפיס את שורת הטקסט הבאה:

```
Westfield Galleria At Roseville
```

e) (6 pts.) Write a piped sequence of commands that assigns a bash variable named `num_stores` with the number of stores whose name consists of exactly two words separated by any number of space characters, and whose first word contains the letter W or w (in upper or lower case). When applied to the example file from page 2, your piped sequence of commands should assign `num_stores` with the value 3. The three stores that satisfy the requirement are: Eastview Mall, Westfield Topanga, and Walden Galleria.

ה) (6 נק') כתבו רצף פקודות משורשרות שמציב במשתנה bash בשם `num_stores` את מספר החנויות ששמן מורכב משתי מילים בדיוק המופרדות ע"י מספר כלשהו של תווי רווח, ושהמילה הראשונה בו מכילה את האות W או w (גדולה או קטנה). כאשר הוא מופעל על הקובץ לדוגמא מעמוד 2, רצף הפקודות המשורשרות שלכם אמור להציב במשתנה `num_stores` את הערך 3. שלושת החנויות העומדות בדרישות הן: Eastview Mall, Westfield Topanga, ו Walden Galleria.

f) **(6 pts.)** Write a piped sequence of commands that prints all store features that appear in the file `lego.txt` together with their counts, sorted from most frequent feature to the least frequent feature. When applied to the example file from page 2, your piped sequence of commands should print the table specified below.

ו) **(6 נק')** כתבו רצף פקודות משורשרות שמדפיס את כל התכונות של חנויות שמוזכרות בקובץ `lego.txt` ביחד עם מספר המופעים שלהן, ממויינות מהתכונה הנפוצה ביותר לנדירה ביותר. כאשר הוא מופעל על הקובץ לדוגמא מעמוד 2, רצף הפקודות המשורשרות שלכם אמור להדפיס את הטבלא הבאה.

```
6 Curbside Pickup
4 Mosaic Maker
2 Minifigure Factory
1 Brick Lab
```

Question 2 (40 points): שאלה 2 (40 נקודות):

► Consider the C source file below named `my_code.c`. The file is missing two function declarations at the top and two function definitions at the bottom.

◄ התבוננו בקובץ הבא בשם `my_code.c` עם קוד בשפת C. הקובץ חסר שתי הכרזות של פונקציות בראשו ושתי הגדרות של פונקציות בתחתיתו.

my_code.c

```
#include <stdio.h>

// here you add declarations functions computeAverage()
// and countDigitsLetters() in (a)

int main() {
     int    array[] = {1, 2, 4, 8, 16, 32};
     char   time[]  = "June 28, 2022, 11AM";
     double average;
     int    digits, letters;

     // set variable average to be the average of
     // array[0], array[1], and array[2] (in this case: 2.333…)
    average = computeAverage(array, 3);

     // set variable digits to the number of digit
     // characters (0-9) in string time (in this case: 8)
     //    -- and --
     // set variable letters to the number of letter
     // characters (a-z & A-Z) in string time (in this case: 6).
     countDigitsLetters(time, &digits, &letters);

     return 0;
}


// here you add definitions to functions computeAverage()
// and countDigitsLetters() in (d) and (e)
```

a) (8 pts.) The top of this source file is missing declarations for the two functions **computeAverage** and **countDigitsLetters** that are invoked by **main**. Write appropriate declarations for these two functions below, considering the way they are invoked in **main** and the documentation provided before each call. Use **const** pointer types when appropriate.

א) (8 נק') בראש קובץ הקוד הזה חסרות הכרזות לשתי הפונקציות **computeAverage** ו **countDigitsLetters** שנקראות על ידי **main**. כתבו למטה הכרזות מתאימות לשתי הפונקציות, כשאתם לוקחים בחשבון את האופן בו קוראים להן מ **main** ואת התיעוד הרשום לפני כל קריאה. השתמשו במצביעי **const** כאשר הדבר מתבקש.

b) (6 pts.) Assume that your solution to (a) above is correct, and you add the correct function declarations to the top of file `my_code.c`. For each of the following two compilation commands specify if it will succeed or not. If it succeeds, specify the name of the generated file and whether it is an executable file or an object file. If the compilation does not succeed, specify the cause for the error and whether it was found in the base compilation stage or the linking stage.

ב) (6 נק') הניחו שהפתרון שלכם לסעיף א' למעלה נכון, ואתם מוסיפים הכרזות תקינות לראש הקובץ `my_code.c`. לכל אחת משתי פקודות הקומפילציה למטה, ציינו אם היא מצליחה או לא. אם היא מצליחה, ציינו את שם הקובץ הנוצר ואם הוא קובץ הרצה או קובץ אובייקט. אם הקומפילציה לא מצליחה, ציינו את הסיבה לשגיאה ואם היא נמצאה בשלב קומפילציית הבסיס או בשלב הקישור.

```
gcc -c my_code.c
```

```
gcc my_code.c
```

c) (10 pts.) For each of the following five invocations of function **countDigitsLetters**, specify whether it will trigger a compilation error, a compilation warning, or will compile without any messages. Provide a brief explanation for each answer. Notice that we highlighted in bold the part of the invocation that is different from the original one.

ג) (10 נק') לכל אחת מחמש הקריאות הבאות לפונקציה **countDigitsLetters**, ציינו אם היא גוררת הודעת שגיאה או הודעת אזהרה בקומפילציה, או שהיא עוברת קומפילציה ללא כל הודעות. ספקו הסבר קצר לכל תשובה. שימו לב שהדגשנו את החלק בקריאה ששונה מהקריאה המקורית.

```
countDigitsLetters(time+1, &digits, &letters);
```

```
countDigitsLetters(time+100, &digits, &letters);
```

```
countDigitsLetters(time, &average, &letters);
```

```
countDigitsLetters(time, &(digits+1), &letters);




countDigitsLetters(time, (&digits)+1, &letters);
```

d) (8 pts.) Write an appropriate definition for function **computeAverage** that fits the description above its invocation in **main** and the declaration that you specified in (a) above.

ד) (8 נק') כתבו הגדרה הולמת לפונקציה **computeAverage** שמתאימה לתיאור שלה מעל הקריאה ב **main**, וגם להכרזה שכתבתם בסעיף א' למעלה.

e) (8 pts.) Write an appropriate definition for function **`countDigitsLetters`** that fits the description above its invocation in **`main`** and the declaration that you specified in (a) above.

ה) (8 נק') כתבו הגדרה הולמת לפונקציה **`countDigitsLetters`** שמתאימה לתיאור שלה מעל הקריאה ב **`main`**, וגם להכרזה שכתבתם בסעיף א' למעלה.

Question 3 (24 points): שאלה 3 (24 נקודות):

► This question considers an implementation for a **list of strings** that uses the generic linked list that we saw in class. Recall that the generic linked list is based on the types defined in the box below. In our case, the `data` field of each list node holds a pointer to the first character of a string.

◄ שאלה זו עוסקת במימוש של **רשימת מחרוזות** שעושה שימוש ברשימה המקושרת הגנרית שראינו בכיתה. זכרו שהרשימה המקושרת הגנרית מבוססת על הטיפוסים המוגדרים בקופסא למטה. במקרה שלנו, השדה `data` של כל חולייה ברשימה מחזיק מצביע לתו הראשון במחרוזת.

```
typedef void* Data;

typedef struct listNode* ListNode;

struct listNode {
   Data data;
   ListNode next;

};
```

► Implement the three functions specified in (a)-(c). Follow these general guidelines:

- Do not use library functions, except where explicitly allowed or required.
- You may assume that the three types above are all “public”. This means that your code may directly access the structure fields.
- Do not use the interface functions we developed in class for the linked list.
- You may assume that every node in a given list holds a valid string.
- There is no need to write documentation.

◄ ממשו את שלושת הפונקציות המתוארות בסעיפים א' – ג'. עקבו אחר ההנחיות הכלליות הבאות:

- אל תיעזרו בפונקציות ספריה, פרט למקרים בהם אתם נדרשים או מורשים לעשות זאת במפורש.
- אתם רשאים להניח ששלושת הטיפוסים למעלה הם "פומביים". כלומר שהקוד שלכם יכול לגשת ישירות לשדות של המבנה.
- אל תעשו שימוש בפונקציות הממשק שפיתחנו בכיתה עבור הרשימה המקושרת.
- אתם רשאים להניח שכל חולייה ברשימה נתונה מכילה מחרוזת תקנית.
- אין צורך בכתיבת תיעוד לקוד.

a) (8 pts.) Implement a function named **`getLongestString`**. The function receives a single parameter (**`head`**) specifying the head of a list of strings. It returns the longest string in the list.

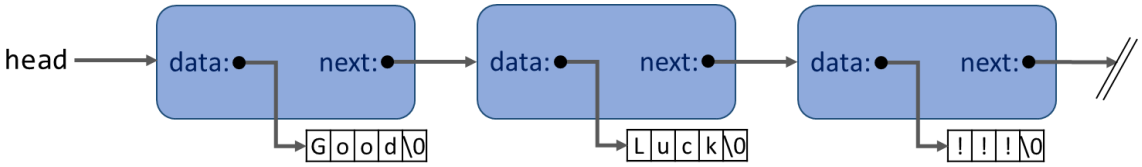
- If the list is empty, the function returns NULL.
- If there is more than one string with the largest length in the list, the function returns the last such string in the list.
- Do not use any library function in your implementation, in particular, no dynamic memory allocation functions or library functions for strings.

For example, when given the head of the list illustrated below, the function returns the string held by the 2nd list node ("Luck").

א) (8 נק') ממשו פונקציה בשם **`getLongestString`**. הפונקציה מקבלת פרמטר אחד (**`head`**) המציין ראש של רשימת מחרוזות. היא מחזירה את המחרוזת הארוכה ביותר ברשימה.

- אם הרשימה ריקה, אז הפונקציה מחזירה NULL.
- אם ישנה ברשימה יותר ממחרוזת אחת בעלת אורך מירבי, אז הפונקציה מחזירה את המחרוזת האחרונה מביניהן ברשימה.
- אל תיעזרו בפונקציות ספריה במימוש שלכם. בפרט, לא בפונקציות הקצאת זיכרון או בפונקציות ספריה למחרוזות.

לדוגמא, כאשר היא מופעלת על ראש הרשימה המצויירת למטה, הפונקציה מחזירה את המחרוזת המוחזקת על ידי החולייה השנייה ("Luck").



```
char* getLongestString(ListNode head) {









}
```

b) (8 pts.) Implement a function named **`popLastString`**. The function receives a single parameter (**`head`**) specifying the head of a list of strings. It pops the last string from the list using the following procedure.

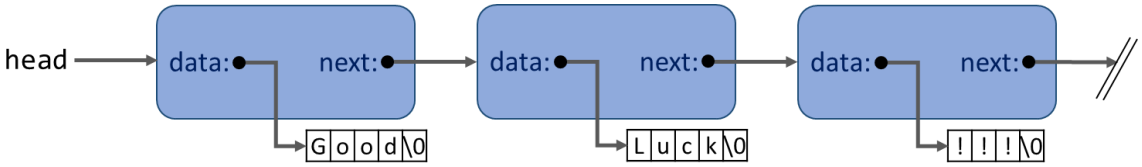
- If the list is empty, the function returns NULL.
- Otherwise, the list is not empty and the function returns the string held by the last node in the list. It also removes and frees that node.
- Use the appropriate functions from `stdlib` for all memory allocation management. See description of relevant functions on page 20.

For example, when given the head of the list illustrated below, the function removes the third list node from the list, frees it, and returns the string held by that node ("!!!").

ב) (8 נק') ממשו פונקציה בשם **`popLastString`**. הפונקציה מקבלת פרמטר אחד (**`head`**) המציין ראש של רשימת מחרוזות. היא מוציאה את המחרוזת האחרונה מהרשימה באופן הבא:

- אם הרשימה ריקה, אז הפונקציה מחזירה NULL.
- אחרת, הרשימה אינה ריקה והפונקציה מחזירה את המחרוזת המוחזקת על ידי החולייה האחרונה ברשימה. היא גם מסירה ומשחררת את החולייה הזו.
- היעזרו בפונקציות המתאימות מ `stdlib` עבור ניהול הקצאת זיכרון דינמי. ראו תיאור של פונקציות רלוונטיות בעמוד 20.

לדוגמא, כאשר היא מופעלת על ראש הרשימה המצויירת למטה, הפונקציה מסירה את החולייה השלישית מהרשימה, משחררת אותה, ומחזירה את המחרוזת שהוחזקה על ידיה ("!!!").



```
char* popLastString(ListNode head) {

}
```

c) (8 pts.) Implement a function named **`mergeTwoNodes`**. The function receives a single parameter (**`head`**) specifying the head of a list of strings. It merges the first two strings in the list using the following procedure.

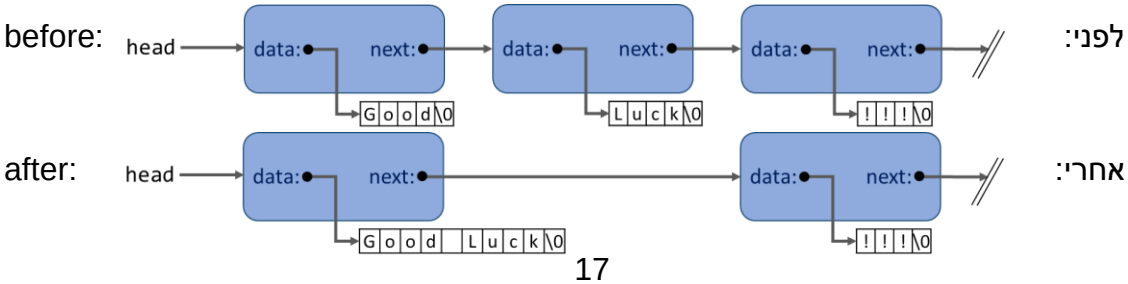
- If the list is empty or contains only one node, the function does nothing and returns 0 (indicating that there aren't two strings to merge).
- Otherwise, the list contains at least two strings, and the function does the following:
  - It concatenates the second string in the list to the first string, with a single space character separating them.
  - It removes the second node from the list and frees it.
  - If the function succeeds in these operations, it returns 1.
  - If the function fails (due to memory allocation failure), it returns 0.
- You should assume here that all strings held by the list are valid (non-NULL pointers) and dynamically allocated (each in a separate block).
- Use the appropriate functions from `stdlib` for all memory allocation management. See description of relevant functions on page 20.
- You may also use library functions from `string`. See description of relevant functions on page 21.

See execution example below.

ג) (8 נק') ממשו פונקציה בשם **`mergeTwoNodes`**. הפונקציה מקבלת פרמטר אחד (**`head`**) המציין ראש של רשימת מחרוזות. היא ממזגת את שתי המחרוזות הראשונות ברשימה באופן הבא:

- אם הרשימה ריקה או שהיא מכילה חולייה בודדת, אז הפונקציה לא עושה דבר ומחזירה 0 (לציין שאין שתי מחרוזות למזג).
- אחרת, הרשימה מכילה לפחות שתי מחרוזות, והפונקציה פועלת כדלהלן:
  - היא משרשרת את המחרוזת השנייה אחרי המחרוזת הראשונה, עם תו רווח בודד מפריד ביניהן.
  - היא מסירה את החולייה השנייה מהרשימה ומשחררת אותה.
  - אם הפונקציה הצליחה בפעולות הנ"ל, היא מחזירה 1.
  - אם הפונקציה נכשלה (בגלל כישלון בהקצאת זיכרון), היא מחזירה 0.
- הניחו שכל המחרוזות שהרשימה מחזיקה הן תקינות (לא NULL) ושהן מוקצות באופן דינמי (כל אחת בבלוק נפרד).
- היעזרו בפונקציות המתאימות מ `stdlib` עבור ניהול הקצאת זיכרון דינמי. ראו תיאור של פונקציות רלוונטיות בעמוד 20.
- אתם גם רשאים להיעזר כאן בפונקציות ספריה מ `string`. ראו תיאור של פונקציות רלוונטיות בעמוד 21.

דוגמאת הרצה להמחשה:



```
int mergeTwoNodes(ListNode head) {

}
```

extra page **דף לכתיבה נוספת**

Functions from standard library stdlib.h

malloc – allocates a new block of memory on the heap

- `void* malloc(size_t num_bytes);`

Function description:
The **malloc**() function allocates a new block of memory of size num_bytes bytes on the heap.

Return value: the start address of the block, or NULL if allocation was not successful.

---

free – frees a block allocated from the heap.

- `void free(void* mem_block);`

Function description:
The **free**() function frees an allocated block of memory on the heap.

Return value: None.

Note: if given an address that is not the beginning address of an allocated block, the function results in runtime error.

---

realloc – changes allocation size of a block allocated on the heap

- `void* realloc(void* mem_block, size_t num_bytes);`

Function description:
The **realloc**() function changes the size of a block that is currently allocated on the heap.

Return value: the start address of the newly-allocated block, or NULL if allocation was not successful.

Notes: location of allocated block may change, and if this happens, its contents are copied to new location. If re-allocation fails, then original block remains allocated in the original size.

Functions from standard library string.h

strlen - calculate the length of a string

- `size_t strlen(const char *s);`

Function description:
The **strlen**() function calculates the length of the string *s*, excluding the terminating null byte ('\0').

Return value:
Function returns the number of bytes (characters) in the string *s*.

---

strcpy, strncpy - copy a string

- **`char *strcpy(char *`***`dest`***`, const char *`***`src`***`);`**
- **`char *strncpy(char *`***`dest`***`, const char *`***`src`***`, size_t `***`n`***`);`**

Function description:

The **strcpy**() function copies the string pointed to by *src*, including the terminating null byte ('\0'), to the buffer pointed to by *dest*. The strings may not overlap, and the destination string *dest* must be large enough to receive the copy. *Beware of buffer overruns!*

The **strncpy**() function is similar, except that at most *n* bytes of *src* are copied. If the length of *src* is less than *n*, **strncpy**() writes additional null bytes to *dest* to ensure that a total of *n* bytes are written.

Warning: If there is no null byte among the first *n* bytes of *src*, the string placed in *dest* will not be null-terminated.

Return value:
Both functions return a pointer to the destination string *dest*
