Exam number: _____ :מספר בחינה

I.D. Number: _____ :מספר ת"ז

Reichman University **אוניברסיטת רייכמן**

Efi Arazi School of Computer Science **בית ספר אפי ארזי למדעי המחשב**

Spring semester – MOED ALEPH, 2023 **סמסטר אביב – מועד א',**

Final Exam – System Programming in C **מבחן מסכם – תכנות מערכות בשפת**

Lecturers: Rani Izsak and Sara H. Geizhals

מרצים: רוני איז'ק, שרה גייצהאלס

Date: July 18, 2023 **תאריך:** 18 ביולי, 2023

Duration: 3 hours. No extensions will be given – plan your time wisely!
Material: Four double-sided pieces of paper with written or typed material
Instructions: All answers should be written **on the exam form only**

משך הבחינה: 3 שעות. לא תינתן הארכת זמן – תכננו את זמנכם בקפידה!
חומר עזר: ארבעה דפים דו-צדדיים של חומר מודפס או כתוב
הנחיות: יש לענות על כל השאלות **על גבי טופס המבחן בלבד**

➔ **Do not submit any draft notebooks** **אין להגיש מחברות טיוטא** ←

Question 1	/30	שאלה 1
Question 2	/14	שאלה 2
Question 3	/6	שאלה 3
Question 4	/50	שאלה 4
Final grade	/100	ציון סופי

Good luck! **בהצלחה!**

Question 1 (30 points): שאלה 1 (30 נקודות):

► A file named `expenses.txt` contains a list of Abby's expenses in the past few days, one line per expense.

◄ קובץ בשם `expenses.txt` מכיל מידע לגבי ההוצאות של אבי בימים האחרונים. כל שורה מייצגת הוצאה אחת.

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

<date><colon>**<expense>**<comma>**<amount>**

- **<date>** specifies the date, with either dashes (`'-'`) or slashes (`'/'`) separating the year (which comes first and has 2 or 4 digits) from the month (which comes second and has 1-2 digits) and the day (which comes third and has 1-2 digits).
- **<expense>** is the description of the expense, and it consists of letters (upper or lower case) and space characters (`' '`).
- **<amount>** is the amount of the expense, and it consists of digits.
- <colon> and <comma> are the colon `':'` and comma `','` 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.

- **<date>** מציין את התאריך, עם dashes (`'-'`) או slashes (`'/'`) המפרידים את השנה (המופיעה ראשונה והנה בעלת 2 או 4 ספרות) מהחודש (המופיע שני והנו בעל 1-2 ספרות) והיום (שמגיע אחרון והנו בעל 1-2 ספרות).
- **<expense>** הנו תיאור ההוצאה. הוא מורכב מאותיות (גדולות או קטנות) ותווי רווח (`' '`).
- **<amount>** הנה כמות ההוצאה והיא מורכבת מספרות.
- <comma> ו <colon> הם תווי הפסיק `','` והנקודתיים `':'`.
- ניתן להניח שכל השורות בקובץ כתובות בפורמט הזה ושאין שורות ריקות בקובץ.

► Contents of the file: `expenses.txt` :תוכן קובץ לדוגמא ◄

```
23-08-30:Groceries,377
2023-09-01:Rent,3000
2023-09-01:Groceries,452
23/09/05:Doctor,33
23/09/05:Rav Kav,190
23-9-06:Groceries,72
2023-9-11:Restaurant,42
```

► For each of the piped sequences of commands specified in (a)-(c) below, write its expected output/action 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\תוצאה הצפייה שלו בהינתן הקובץ לדוגמא למעלה. אתם רשאים להוסיף הסבר קצר לתשובה שלכם.

a) (5 pts.) **א) (5 נק')**

```
head -n1 expenses.txt | tr -d '[:digit:]-:,e'
```

Expected **output**: :פלט צפוי

b) (5 pts.) **ב) (5 נק')**

```
paste expenses.txt expenses.txt -d "*" | \
     tail -n1 >> expenses.txt
```

Expected **action**: :תוצאה הצפייה

c) (6 pts.) **ג) (6 נק')**

```
cat expenses.txt | cut -c2,4 | uniq
```

Expected **output**: פלט צפוי:

► Solve the tasks in (d)-(e) as instructed. Your code should work on every input file `expenses.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'-ה על פי ההוראות. הקוד שלכם אמור לעבוד על כל קובץ קלט `expenses.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 to the console screen the expenses under 100 in the order in which they appear in the file. When applied to the example file from page 2, your piped sequence of commands should print `33`, `72`, and `42` (one per line).

ד) (6 נק') כתבו רצף פקודות משורשרות שמדפיס את ההוצאות הנמוכות מ-100, באותו הסדר בו הן מופיעות בקובץ. כאשר הוא מופעל על הקובץ לדוגמא מעמוד 2, רצף הפקודות המשורשרות שלכם אמור להדפיס את המספרים: `33`, `72`, `42` (אחד בכל שורה).

e) (8 pts.) Write a piped sequence of commands to standardize the dates into the more unified format of m/d/yyyy.

This means that: the month should be written with 1 digit (e.g., 8 – and not 08 – for August), and only with 2 digits if needed (e.g., 10 for October); the day should be written with 1 digit (3 for the 3rd of the month), and only with 2 digits if needed (for the 10th of the month and on), and the year should be written with four digits (2023 for this year). The month, day, and year are separated by slashes. Put the output into a file called `output.txt` in the same directory. (You may assume there is no file called `output.txt` in the directory.)

ה) (8 נק') כתבו רצף פקודות משורשרות המשנה את פורמט התאריכים לפורמט m/d/yyyy.

כלומר, החודש אמור להיות מיוצג ע"י ספרה אחת בלבד, אלא אם כן נדרשות שתי ספרות בכדי לייצגו (כלומר, 8 עבור אוגוסט, ולא 08, אבל 10 עבור אוקטובר). היום אמור להיות מיוצג ע"י ספרה אחת בלבד, אלא אם כן נדרשות שתי ספרות בכדי לייצגו (כלומר, 3 עבור 3 בחודש כלשהו, ולא 03, אבל 10 ל-10 בחודש כלשהו). השנה תמיד תיוצג באמצעות 4 ספרות בדיוק. ההפרדה הנה באמצעות slashes. הפלט צריך להיות לתוך קובץ בשם `output.txt` באותו הדירקטורי (ניתן להניח שאין קובץ כזה באותו הדירקטורי).

Question 2 (14 points): שאלה 2 (14 נקודות):

a) (12 pts.) Write an executable bash script `max_words.sh` that prints the count of the greatest number of words on lines 2-3 of all files in a specified directory with a filename that begins with "file." Exact instructions are as follows:

- The script requires exactly 1 input: a directory `dir`. If there is not exactly 1 input, then exit with 1.
- Then the script confirms that the directory `dir` exists. If it does not, then exit with 2.
- Then: get a list of all files in `dir` whose filename begins with `file`. Iterate through those files and for each file make sure that it is a file (and not a directory) and if so then count the number of words on lines 2-3 of the file. (You may assume each file has at least 3 lines.)
- Determine the maximum number of words and print that to the screen.
- Make sure your bash script is executable.

א) (12 נק') כתבו קובץ bash script בשם `max_words.sh` העושה את הפעולה הבאה. נסתכל על השורות השנייה והשלישית (2-3) בכל הקבצים בדירקטורי נתון עם שם קובץ המתחיל ב-"file". נדפיס את מספר המילים הגדול ביותר בשורות הללו מבין כל הקבצים בדירקטורי (רק את ה-COUNT). להלן הוראות מפורטות:

- הסקריפט יקבל קלט אחד בדיוק – דירקטורי dir. אם לא יהיה קלט (פרמטר) אחד בדיוק – הוא ייצא עם קוד 1.
- אז, הסקריפט יוודא שהדירקטורי אכן קיים. אם לא – הוא ייצא עם קוד 2.
- אם הכל בסדר (כלומר, הסקריפט לא יצא עם קוד 1 או עם קוד 2), הסקריפט ימצא את כל הקבצים ב-dir ששם הקובץ שלהם מתחיל ב-file, יעבור עליהם, ולכל אחד מהם יוודא שהוא אכן קובץ רגיל (ואינו דירקטורי). אם אכן כך, הסקריפט יספור את מספר המילים בשורות 2-3 בקובץ זה (הניחו שלקובץ אכן ישנן 3 שורות, לפחות).
- הסקריפט ימצא את מספר המילים המקסימלי מבין המילים המתוארות לעיל (שורות 2-3 בקבצים המתחילים ב-file בדירקטורי הנתון) וידפיס את המספר הנ"ל למסך.
- יש לוודא שה-BASH SCRIPT הנו EXECUTABLE.

(2 pts.) Assume that your solution to (a) above is correct. Currently, your permissions for `max_words.sh` is set to `-rw-r--r--`; write the command that will set your permissions so that you, as the user, will be able to run the executable file.

ב) (2 נק') הניחו שהפתרון שלכם לסעיף א' למעלה נכון. נכון לעתה, ההרשאות שלכם ל`max_words.sh` - הנן `-rw-r--r--` כתבו את הפקודה שתשנה את ההרשאות כך שתוכלו כ-USER להריץ את קובץ ה-EXECUTABLE שכתבתם.

Question 3 (6 points): שאלה 3 (6 נקודות):

► Examine the C program below and write the expected output of lines A-C in the provided space. Write “undetermined” if you think that a certain printed value is undetermined by the source code. You may assume that the code is successfully compiled and executed on our course server (`sysprog.runi.ac.il`).

◄ בחנו את התוכנית הבאה ב C וכתבו את הפלט הצפוי שלהן בשורות A-C במקום המיועד לכך. כתבו "לא ידוע" אם אתם חושבים שערך מודפס מסוים לא נקבע על ידי הקוד של התוכנית. אתם רשאים להניח שהקוד עובר קומפילציה בהצלחה ורץ על שרת הקורס שלנו (`sysprog.runi.ac.il`).

```
#include <stdio.h>

int main() {
  int array[] = {1, 0, 2, 4};
  int i = sizeof(arr)*10;
  printf("%d\n", array[i]);          //line A

  printf("%d\n", 40&(3<<4));         //line B

  char str[30] = "moed aleph";
  str[1]++;
  printf("%s\n", str);               //line C

  return 0;
}
```

► Write here the expected output. No need to explain your answers.

◄ כתבו כאן את הפלט הצפוי. אין צורך להסביר את תשובתכם.

Line A:

Line B:

Line C:

Question 4 (50 points): שאלה 4 (50 נקודות):

► This question involves implementing functions for the linked list data structure that we discussed in class. In particular, we consider a linked list of `double` values based on the types defined in the box below. Recall that we use `Data` to encapsulate the type of the values (`double` in this case). The structure type is used to represent a list node, which holds a data value (of type `Data`) and a pointer to the next node in the list. A list is represented by a pointer to the first node (head), and an empty list is represented by a NULL pointer.

◄ השאלה הזאת עוסקת במימוש פונקציות למבנה הנתונים של רשימה מקושרת שראינו בכיתה. בפרט, אנחנו מתייחסים לרשימה מקושרת של ערכי `double` המבוססת על הטיפוסים המוגדרים במסגרת למטה. כזכור לכם, אנחנו משתמשים ב `Data` להתייחס לטיפוס של הערכים (`double` במקרה הזה). טיפוס המבנה משמש לייצג חוליה ברשימה, שמחזיקה את ערך הנתון (data) מטיפוס `Data` ומצביע לחוליה הבאה ברשימה (next). רשימה מיוצגת על ידי מצביע לחוליה הראשונה ברשימה (ראש הרשימה – head), ורשימה ריקה מיוצגת על ידי מצביע NULL.

```
typedef double Data;
typedef struct listNode* ListNode;
struct listNode {
   Data data;
   ListNode next;
};
```

► Implement the five functions specified below in (a)-(e). Follow these general guidelines:

- You are only allowed to use standard library functions for dynamic memory allocation. A brief description of the relevant functions is provided on page 21.
- You cannot assume that you have access to any other linked list function, including those we implemented in class. You need to write your implementation from scratch.
- There is no need to write documentation.

◄ ממשו את חמש הפונקציות המתוארות למטה בסעיפים א' – ה'. עקבו אחר ההנחיות הכלליות הבאות:

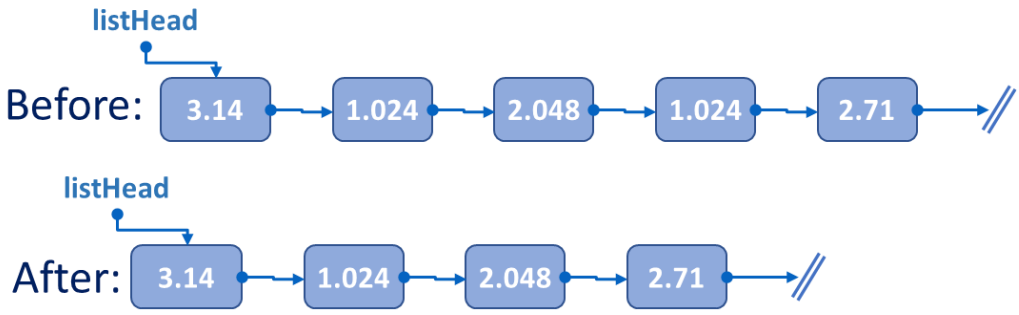
- אתם רשאים להשתמש בפונקציות ספריה סטנדרטיות רק לצורך הקצאה דינמית של זיכרון. תיאור תמציתי של הפונקציות הרלוונטיות מסופק לכם בעמוד 21.
- אתם לא יכולים להניח שיש לכם גישה לפונקציות אחרות עבור רשימה מקושרת, כולל אלה שמימשנו בכיתה. אתם צריכים לכתוב מימוש מאפס.
- אין צורך בכתיבת תיעוד לקוד.

a) (10 pts.) Implement a function named **`deleteSmallestNode`**. The function receives as a parameter a linked list (**`listHead`**). It then returns the same list but without the node that holds the smallest value. Follow these guidelines:

- If the list is empty, the function does nothing and returns the list as is.
- If the list is not empty, the function removes the list node with the smallest value from the list and it frees the memory that this node occupied.
- If there are several nodes in the list that hold the smallest value, then the function removes the last of these nodes in the list. See example below.
- The function returns a pointer to the head of the updated list. Pay attention to the special case when the first node in the list is the one being removed.

א) (10 נק') ממשו פונקציה בשם **`deleteSmallestNode`**. הפונקציה מקבלת כפרמטר רשימה מקושרת (**`listHead`**), והיא מחזירה את הרשימה ללא החולייה שמחזיקה את האיבר בעל הערך הנמוך ביותר. עקבו אחר ההנחיות הבאות:

- אם הרשימה ריקה, הפונקציה לא עושה דבר ומחזירה את הרשימה כפי שהיא.
- אם הרשימה אינה ריקה, הפונקציה מסירה את החולייה ברשימה בעלת הערך הנמוך ביותר ומשחררת את הזיכרון שהחולייה הזו תפסה.
- אם ישנן מספר חוליות ברשימה בעלות הערך הנמוך ביותר, אז הפונקציה מסירה את האחרונה מביניהן ברשימה. ראו דוגמא למטה.
- הפונקציה מחזירה מצביע לראש הרשימה המעודכנת. שימו לב למקרה המיוחד שבו החולייה הראשונה ברשימה היא זו שמוסרת על ידי הפונקציה.



```
ListNode deleteSmallestNode(ListNode listHead) {









}
```

b) (10 pts.) Implement a function named **`listToArray`**. The function receives as a parameter a linked list (**`listHead`**), and it returns a dynamically allocated array that holds the same values that the list holds (in the exact same order). Follow these guidelines:

- If the list is empty, the function does nothing and returns NULL.
- If the list is not empty, the function allocates the exact amount of memory required for the new data array, and then sets the array entries according to the appropriate data values in the list.
- If allocation failed, the function returns NULL. Make sure to avoid memory leaks.
- Otherwise, the function returns a pointer to the newly created data array.

ב) (10 נק') ממשו פונקציה בשם **`listToArray`**. הפונקציה מקבלת כפרמטר רשימה מקושרת (**`listHead`**), והיא מחזירה מערך מוקצה דינמית שמכיל את אותם ערכים שהרשימה מחזיקה (בדיוק באותו הסדר). עקבו אחר ההנחיות הבאות:

- אם הרשימה ריקה, הפונקציה לא עושה דבר ומחזירה NULL.
- אם הרשימה אינה ריקה, הפונקציה מקצה את כמות הזיכרון המדויקת הדרושה עבור מערך הנתונים החדש, והיא קובעת את תאי המערך לפי ערכי הנתונים המתאימים מהרשימה.
- אם ההקצאה לא מצליחה, אז הפונקציה מחזירה NULL. שימו לב להימנע מדליפות זיכרון.
- אחרת, הפונקציה מחזירה מצביע למערך הנתונים החדש שהיא יצרה.

```
Data* listToArray(ListNode listHead) {




}
```

c) (10 pts.) Implement a function named **`arrayToList`**. The function receives as parameters an array of data values (**`dataArray`**), and a number of elements (**`n`**). It creates a dynamically allocated linked list with **`n`** list nodes that hold the first **`n`** values that the array holds in the exact same order. Follow these guidelines:

- If **`n`**≤ 0, the function returns an empty list.
- If **`n`**>0, the function allocates memory for all list nodes, sets their values, and connect them according to the order of values in the input array.
- If any memory allocation failed, the function returns NULL. Make sure to avoid memory leaks.
- Otherwise, the function returns a pointer to the head of the new list.

ג) (10 נק') ממשו פונקציה בשם **`arrayToList`**. הפונקציה מקבלת כפרמטרים מערך של ערכי נתונים (**`dataArray`**), ומספר של איברים (**`n`**). היא יוצרת רשימה מקושרת מוקצית דינמית בעלת **`n`** חוליות שמחזיקה את **`n`** הערכים הראשונים שהמערך מחזיק, בדיוק באותו הסדר. עקבו אחר ההנחיות הבאות:

- אם 0≤**`n`**, הפונקציה מחזירה רשימה ריקה.
- אם 0<**`n`**, הפונקציה מקצה זיכרון לכל חוליות הרשימה, קובעת את הערכים שלהן, ומחברת אותן לפי הסדר של הערכים במערך הקלט.
- אם הקצאה כלשהי של זיכרון נכשלה, הפונקציה מחזירה NULL. שימו לב להימנע מדליפות זיכרון.
- אחרת, הפונקציה מחזירה מצביע לראש הרשימה החדשה

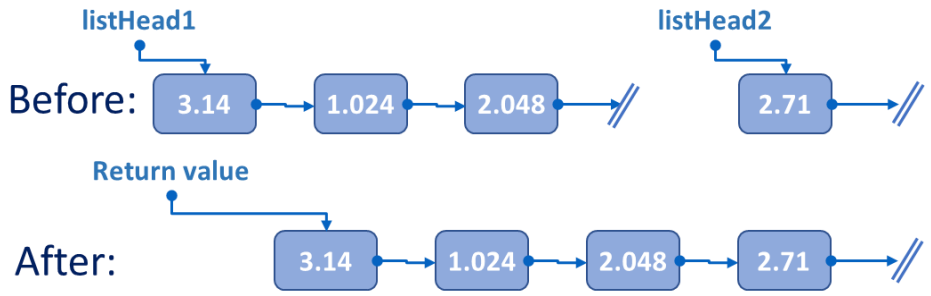
```
ListNode arrayToList(Data* dataArray, int n) {




}
```

d) (10 pts.) Implement a function named **`concatLists`**. The function receives as parameters two linked lists (**`listHead1`**, **`listHead2`**), and it concatenates the second list to the end of the first one. Follow these guidelines:

- The function does not create new list nodes. It just changes connections between the existing list nodes. See example below.
- The function returns a pointer to the head of the concatenated list.
- Make sure to consider the special cases when one of the lists is empty.

ד) (10 נק') ממשו פונקציה בשם **`concatLists`**. הפונקציה מקבלת כפרמטר שתי רשימות מקושרות (**`listHead1`**, **`listHead2`**) והיא משרשרת את הרשימה השנייה לסוף הרשימה הראשונה. עקבו אחר ההנחיות הבאות:

- הפונקציה לא יוצרת חוליות חדשות ברשימה. היא רק משנה את הקישורים בין החוליות הקיימות ברשימות. ראו דוגמא למטה.
- הפונקציה מחזירה מצביע לראש הרשימה המשורשרת.
- שימו לב להתייחס למקרים המיוחדים בהם אחת הרשימות ריקה.



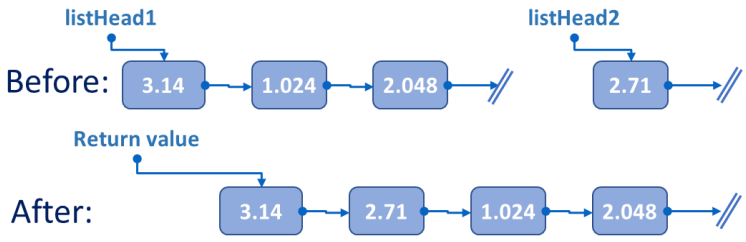
```
ListNode concatLists(ListNode listHead1, ListNode listHead2) {




}
```

e) (10 pts.) Implement a function named **`mergeLists`**. The function receives as parameters two linked lists (**`listHead1`**, **`listHead2`**), and it merges them in an alternating fashion, as described below:

- Let n denote the length of the shorter of the two input lists. The first 2n nodes in the merged list are set as follows:
  - ► The 1st, 3rd, …, (2n-1)th nodes in the merged list are the first n nodes in the first input list (**`listHead1`**), in order.
  - ► The 2nd , 4th , …, (2n)th nodes in the merged list are the first n nodes in the second input list (**`listHead2`**), in order.
- If one of the lists is longer than the other (longer than n), then the remaining nodes of the merged list (beyond the 2nth node), are the remaining nodes in the longer list (beyond the nth node).
- The function does not create new list nodes. It just changes connections between the existing list nodes. See example below.
- You should return the head of the merged list.
- Make sure to consider the special cases when one of the lists is shorter than the other, or when one of them is empty.

ה) (10 נק') ממשו פונקציה בשם **`mergeLists`**. הפונקציה מקבלת כפרמטר שתי רשימות מקושרות (**`listHead1`**, **`listHead2`**), והיא ממזגת את שתי הרשימות לסירוגין באופן הבא:

- נסמן ב n את אורך הרשימה הקצרה מבין שתי רשימות הקלט. אז 2n החוליות הראשונות ברשימה הממוזגת נקבעות כך:
  - החוליות הראשונה, השלישית, ... ה (2n-1) ברשימה הממוזגת הן n החוליות הראשונות ברשימת הקלט הראשונה (**`listHead1`**), בסדר הזה.
  - החוליות השנייה, הרביעית, ... ה (2n) ברשימה הממוזגת הן n החוליות הראשונות ברשימת הקלט השנייה (**`listHead2`**), בסדר הזה.
- אם אחת מרשימות הקלט ארוכה יותר מהשנייה (ארוכה יותר מ n), אז החוליות הנותרות ברשימה הממוזגת (מעבר לחולייה ה 2n), הן החוליות הנותרות ברשימה הארוכה יותר (מעבר לחולייה ה n).
- הפונקציה לא יוצרת חוליות חדשות ברשימה. היא רק משנה את הקישורים בין החוליות הקיימות ברשימות. ראו דוגמא למטה.
- הפונקציה מחזירה מצביע לראש הרשימה הממוזגת.
- שימו לב להתייחס למקרים המיוחדים בהם אחת הרשימות קצרה יותר מהשנייה, או שאחת מהן ריקה.



```
ListNode  mergeLists(ListNode listHead1, ListNode listHead2) {









}
```

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*
