תרגיל בית 1

- בתרגיל זה אין להשתמש בהקצאות זיכרון דינמיות.

<u>הנחיות כלליות לפיתרון התרגילים בקורס:</u>

- התרגילים הם לעבודה ביחידים. **מותר להתייעץ אך ורק בעל פה, אסור בתכלית האיסור שחומר כתוב/מודפס/אלקטרוני יעבור בין אנשים. בנוסף, על חלק מהתרגילים תיבחנו פרונטלית ועליכם להבין כל דבר בקוד!**
- המנעו ממספרי קסם: מספרים שמופיעים באמצע הקוד בלי משמעות מיוחדת (לדוגמא נניח שמספר הרשומות בתרגיל אחר הוא מקסימום 50 ואז בכל מקום בקוד כתוב 50. לעומת זאת, 0 לתחילת מערך לא נחשב מספר קסם - הפעילו הגיון בריא) והשתמשו במקום זאת בפקודות מאקרו (#define או אם כבר למדתם על כך ב const).
- אין להשתמש ב-variable length arrays , וכן במשתנים סטאטיים ו / או גלובליים.
- כל התרגילים בקורס צריכים להתקמפל ולרוץ באתר c9.io (האתר מריץ מערכת הפעלה אובונטו) עם שורת הקמפול:

  gcc -Wall -Wvla -Werror -g ...

  עבור תרגילי C ושורת הקמפול:

  g++ -Wall -Wvla -Werror -g -D_GLIBCXX_DEBUG -std=c++11 ...

  עבור תרגילי C++
- או במידה ומצורף Makefile עם ה Makefile המצורף
- יש להקפיד על סגנון תכנות טוב כמו שלמדתם. לדוגמא, להימנע מחזרות קוד (לכתוב פונקציות שצריך), שמות משתנים עם משמעות, בהירות הקוד, תיעוד הקוד, להקפיד להשתמש בקבועים שצריך ולא במספרי קסם וכו'.
- עליכם להגיש קובץ ששמו מספר ת.ז. שלכם כמו שהיא מופיעה באתר המודל נקודה zip. לדוגמא, אם מספר ת.ז. שלי הוא 12345678 אז שם הקובץ יהיה:

  12345678.zip

  השתמשו ב make zipfile ע"מ לייצר קובץ תקין ו make checkzipfile ע"מ לבדוק אותו
  חשוב: חלק מהבדיקה אוטומטית ואי עמידה בקמפול/בדיקה ע"י ה Makefile הניתן יגרור אפס מיידית.

An ASCII table can be found at the site www.asciitable.com. To solve these programming problems use as needed the library functions for input/output: getchar, scanf, printf and others. The specifications for these and other, input/output functions can be found here: http://www.cplusplus.com/reference/clibrary/cstdio/.
In order to use the functions specified in the above link one must add at the beginning of the program the macro:
#include <stdio.h>
It is also recommended to use the library math functions. Details of these functions can be found here:
http://www.cplusplus.com/reference/clibrary/cmath/
Please take note of the following:

In this exercise one must not use a data structure by using the *struct* keyword or arrays or dynamic memory allocations. Solutions that use the *struct* keyword or arrays or dynamic memory allocation will be disqualified.
In the examples shown in this exercise **boldface type** indicates that this is input that the user enters at the keyboard. Text not in boldface type is printed result of the program.
Examples of input and output are available on the moodle under the link for this file.
Remark:
Although input/output is from the shell, it is recommended to check yourself with files (see fileReadingExample.c or use piping in linux shell (< to redirect input > to redirect output).

1. (30 points)
Write a program named CaseChange.c. This program will get from the keyboard a single alphanumeric character A-Z, a-z and 0-9 (from the ASCII table). If the character entered is not in the range listed then it is invalid. The program terminates immediately after the output is displayed on the screen. The input/output scenarios for different types of characters are as follows:
If the character entered is an upper case English letter, then the program must print the input on the following line and print and arrow to the right of the input. To the right of the arrow the lower case version of the letter is to be printed, followed by a newline character (\n).

B
B->b

If the character entered is a lower case English letter then the program must print the input on the following line and print and arrow to the right of the input. To the right of the arrow the upper case version of the letter is to be printed, followed by a newline character (\n).

a
a->A

If the character entered is a digit (0-9) then the program must print on the following line the square of the input digit, followed by a newline character (\n).
7
49

In the event that the input is not in one of these sets of characters (0-9, A-Z, a-z) then an arrow is printed to the right of the input flowed by the string "Invalid Input" followed by the newline character (\n).

/
/->Invalid Input

In all of the i/o scenarios listed above the program terminates after a single i/o sequence. Please note that there are relevant library functions in the header file ctype.h, however in this exercise you are not allowed to use these functions**.**
Do not use Magic numbers, i.e, unique values with unexplained meaning or multiple occurrences). Use named constants (using #define we will later learn about const). Please name the code file CaseChange.c.
Hint – Look at the value relationship between upper case and lower case English letters in the ASCII table.

2. (70 Points)
Write a program named DrawPolynomial. This program shall print a 3rd degree polynomial to the shell screen within a predefined area of the screen (on the x axis in the range ±35 and on the y axis in the range of ±10. The center of the screen is at the point (0,0) and each unit on both the x and y axis is a space or line in the shell screen(a movement of a single space or newline in the non graphical shell screen) for the respective axes. The values shown above for the limits of the drawing on the screen shall be clearly defined in the code and be easily changeable. The axes shall be drawn with lines and the polynomial drawn using the "*" character. After program invocation the prototypical 3rd degree polynomial is printed on the screen, followed by an input
sequence in which the coefficients of the polynomial are entered at the keyboard:
>DrawPolynomial
y(x) = a + b*x + c*x^2 + d*x^3
Select a:
-3
Select b:
24
Select c:
-5.345
Select d:
1.23
You can assume that the input is numeric and can define these variables (a-d) as type double.
After the above data entry, the function with the data entered is displayed
y(x) = (-3) + 24*x + (-5.345)*x^2 + 1.23*x^3
The coefficients of the polynomial can have up to 3 places to the right of the decimal point. printf enables printing such a number using the formatting delimiter "%.3f". After printing the polynomial the graph is displayed on the shell screen. Since the printing is done one character at a time and **the use of arrays is not allowed in this exercise**, one must compute for each value of (x,y) in the range shown above whether the polynomial "passes" through that point(considering the y range also) or the y value is close enough(±½). (Y is the value of the polynomial). Below is an example of a graph of the function x3+1 in accordance to the graphing scheme shown below.
Printing Format (see also examples at moodle)

In each of the 21 rows (-10<= y <= 10) 71 characters will be printed corresponding to the values of x from -35 to +35(including zero). At the end of the row a newline character is printed ("\n").
The printing is to be done in the following manner (printing to be done by rows):
If for the current value of x the polynomial value the y value is close (< ±½) then the character '*' is printed.
If for the current value of x the polynomial value the y value is not close (>= ±½) then a blank character is printed.(if the (X,Y) pair is not on the axes
At the origin (0,0) and the polynomial does not pass there display the'+' character.
If the polynomial does not pass through a point which is on the x axis display the '-' character.
If the polynomial does not pass through a point which is on the y axis display the '|' character. Please name the code file DrawPolynomial.c. For Illustrations see the .out files in ex1_test_files/DrawPolynomial. It is advised to view these files with a text editor that can display at least 72 characters per row (emacs or Notepad++ for example) or to use linux 'cat' command.

בהצלחה!