24-780 Engineering Computation Problem Set 02

You need to create a ZIP file (It may appear as a compressed folder in Windows) and submit the ZIP file via the 24-780 Canvas course. The file name of the ZIP file must be:

PS02-YourAndrewID.zip

For example, if your Andrew account is *hummingbird@andrew.cmu.edu*, the file name must be:

PS02-hummingbird.zip

If your ZIP file does not comply with this naming rule, you will automatically lose 5% credit from this assignment. If we are not able to identify who submitted the file, you will lose another 5% credit. If we finally are not able to connect you and the submitted ZIP file, you will receive 0 point for this assignment. Therefore, please make sure you strictly adhere to this naming rule before submitting a file.

The ZIP file needs to be submitted to the 24-780 Canvas course. If you find a mistake in the previous submission, you can re-submit the ZIP file with no penalty as long as it is before the submission deadline.

Notice that the grade will be given to the final submission only. If you submit multiple files, the earlier version will be discarded. Therefore, if you re-submit a ZIP file, the ZIP file MUST include all the required files. Also, if your final version is submitted after the submission deadline, late-submission policy will be applied no matter how early your earlier version was submitted.

Make sure your program can be compiled with no error in one of the compiler servers. Don't wait until the last minute. Compiler servers may get very busy minutes before the submission deadline!

The ZIP file needs to include:

- C++ source file of your program (ps2-1.cpp)
- C++ source file of your program (ps2-2.cpp)

Double check that your file name of your Zip file and source files comply with the specification. TAs grade your assignments with a grading script, but the script may not find your submission if the file name is not compliant, and the TA will need to stop the script and manually correct your file name(s). You won't want to be graded by a frustrated TA on top of 5% point deduction!

Submission Due: Please see Canvas

START EARLY!

Unless you are already a good programmer, there is no way to finish the assignment overnight.

PS2-1 Sine Wave on the Console Window[ps2-1.cpp] (30 pts)

Write a function called:

void Hash(int x);

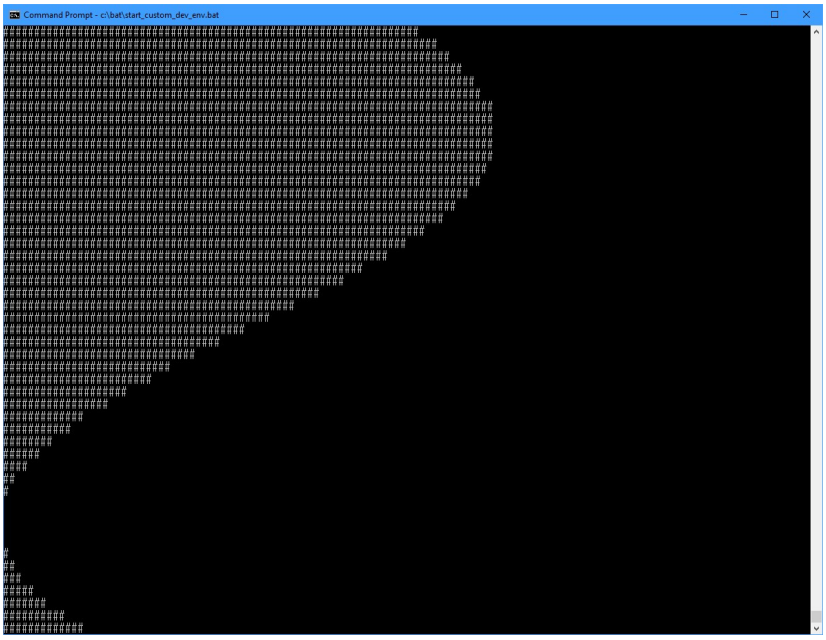
which prints x hash marks (#) and a line break (\n or std::endl).

In main function, run a loop on a double-precision floating point from zero to 100 with 0.1 increment. Inside the loop, call Hash function to draw (40+sin(a)*40) hash marks.

Include appropriate header files. You can choose C standard library or C++ Standard Template Library.

The program will render a sine wave on the console window.

Make sure your program can be compiled with no error in one of the compiler servers. Don't wait until the last minute. Compiler servers may get very busy minutes before the submission deadline!



Sample Output (Expand the window so that it at least prints 80 characters in a row).

PS2-2 Digital Flashcards of Multiplication [ps2-2.cpp] (70 points)

In this problem, you will write a C++ code that serves as a learning tool for elementary multiplication from 1x1 to 12x12. Your C++ code should be a console application and roughly mimics the functionality of a conventional paper flashcard, shown in the figure below.

A flashcard is any of a set of cards bearing information, as words or numbers, on either or both sides, used in classroom drills or in private study. Flashcards can bear vocabulary, historical dates, formulas or any subject matter that can be learned via a question and answer format. Flashcards are widely used as a learning drill to aid memorization by way of spaced repetition.

Your C++ console flash card should:

- First ask the user how many flash cards to work on. The number needs to be between 1 and 144. If the user requests less than 1 or greater than 144 flash cards, show a message (eg. "The number of cards must be between 1 and 144.") and prompt the user to re-enter a different number.
- Show questions one by one on the console window and prompt the user to type the answer. If the user types a wrong answer, tell the user that the answer is wrong and show the correct answer then move on to the next card. If the answer is correct, move on to the next card.
- When all the flash cards are answered, display a feedback comment that includes:
  - The number of problems,
  - Time required to complete the problems in the number of seconds
  - The number of correct answers and percent that the user answered correctly.

  For example,

  You answered 25 problems in 80 seconds.

  You answered 20 problems correctly (80%).

  The time for completion does not have to be sub-second accurate. It can have plus-minus 1 second error. Make sure you measure real time, not CPU time in any environment.

- Problems should be selected and presented randomly, but no same question should appear more than once. Make sure to use srand so that the order of the card appearance changes time to time.

Write the C++ program (named ps2-2.cpp). For full credit:

- Make shuffling a separate function (and use it).
- Also make a separate function for swapping two integers and use it from the shuffling function.

The C++ source code must be included in the Zip file that you submit to the Canvas.

Make sure your program can be compiled with no error in one of the compiler servers. Don't wait until the last minute. Compiler servers may get very busy minutes before the submission deadline!

Hint: You really need one integer per card.

#0 1x1=	#1 1x2=	#2 1x3=	#3 1x4=	#4 1x5=	#5 1x6=	#6 1x7=	#7 1x8=	#8 1x9=	#9 1x10=	#10 1x11=	#11 1x12=
#12 2x1=	#13 2x2=	#14 2x3=	#15 2x4=	#16 2x5=	#17 2x6=	#18 2x7=	#19 2x8=	#20 2x9=	#21 2x10=	#22 2x11=	#23 2x12=
#24 3x1=	#25 3x2=	#26 3x3=	#27 3x4=	#28 3x5=	#29 3x6=	#30 3x7=	#31 3x8=	#32 3x9=	#33 3x10=	#34 3x11=	#35 3x12=
#36 4x1=	#37 4x2=	#38 4x3=	#39 4x4=	#40 4x5=	#41 4x6=	#42 4x7=	#43 4x8=	#44 4x9=	#45 4x10=	#46 4x11=	#47 4x12=
#48 5x1=	#49 5x2=	#50 5x3=	#51 5x4=	#52 5x5=	#53 5x6=	#54 5x7=	#55 5x8=	#56 5x9=	#57 5x10=	#58 5x11=	#59 5x12=
#60 6x1=	#61 6x2=	#62 6x3=	#63 6x4=	#64 6x5=	#65 6x6=	#66 6x7=	#67 6x8=	#68 6x9=	#69 6x10=	#70 6x11=	#71 6x12=
#72 7x1=	#73 7x2=	#74 7x3=	#75 7x4=	#76 7x5=	#77 7x6=	#78 7x7=	#79 7x8=	#80 7x9=	#81 7x10=	#82 7x11=	#83 7x12=
#84 8x1=	#85 8x2=	#86 8x3=	#87 8x4=	#88 8x5=	#89 8x6=	#90 8x7=	#91 8x8=	#92 8x9=	#93 8x10=	#94 8x11=	#95 8x12=
#96 9x1=	#97 9x2=	#98 9x3=	#99 9x4=	#100 9x5=	#101 9x6=	#102 9x7=	#103 9x8=	#104 9x9=	#105 9x10=	#106 9x11=	#107 9x12=
#108 10x1=	#109 10x2=	#110 10x3=	#111 10x4=	#112 10x5=	#113 10x6=	#114 10x7=	#115 10x8=	#116 10x9=	#117 10x10=	#118 10x11=	#119 10x12=
#120 11x1=	#121 11x2=	#122 11x3=	#123 11x4=	#124 11x5=	#125 11x6=	#126 11x7=	#127 11x8=	#128 11x9=	#129 11x10=	#130 11x11=	#131 11x12=
#132 12x1=	#133 12x2=	#134 12x3=	#135 12x4=	#136 12x5=	#137 12x6=	#138 12x7=	#139 12x8=	#140 12x9=	#141 12x10=	#142 12x11=	#143 12x12=

Exercise: You do not submit anything for the exercise, and will not be penalized for not working on the exercise. Just for your practice.

- Re-write Shuffling algorithm using Range-Based For loop.