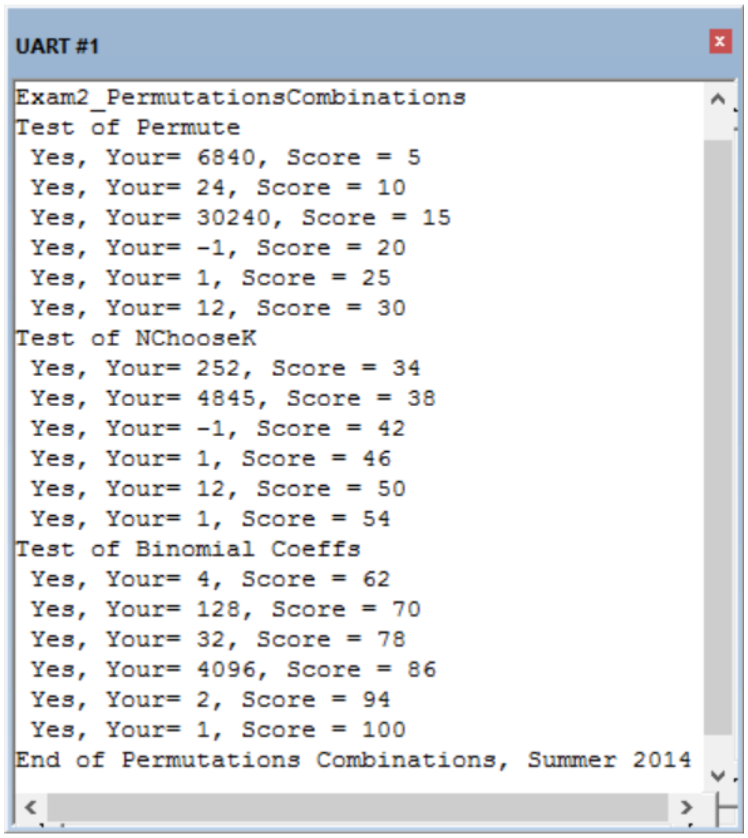
First:________________ Last:______________________

Scoring The correct output values are shown in the figure on the right. Your grade will be based both on the numerical results returned by your program and on your programming style. In particular, write code that is easy to understand, easy to debug, easy to change. Please employ good labels, pretty structure, and good comments.

Performance Score= Run by TA at the checkout		TA:

I promise to follow these rules

This is a closed book exam. You must develop the software solution using the **Keil uVision** simulator. You have 70 minutes, so allocate your time accordingly. You must bring a laptop and are allowed to bring only some pens and pencils (no books, cell phones, hats, disks, CDs, or notes). You will have to leave other materials up front. Each person works alone (no groups). You have full access to **Keil uVision**, with the **Keil uVision** help. You may use the Window's calculator. You sit in front of a computer and edit/assemble/run/debug the programming assignment. You do NOT have access the book, internet or manuals. You may not take this paper, scratch paper, or rough drafts out of the room. You may not access your network drive or the internet. You are not allowed to discuss this exam with other EE319K students until Friday afternoon.



```
Exam2_PermutationsCombinations
Test of Permute
 Yes, Your= 6840, Score = 5
 Yes, Your= 24, Score = 10
 Yes, Your= 30240, Score = 15
 Yes, Your= -1, Score = 20
 Yes, Your= 1, Score = 25
 Yes, Your= 12, Score = 30
Test of NChooseK
 Yes, Your= 252, Score = 34
 Yes, Your= 4845, Score = 38
 Yes, Your= -1, Score = 42
 Yes, Your= 1, Score = 46
 Yes, Your= 12, Score = 50
 Yes, Your= 1, Score = 54
Test of Binomial Coeffs
 Yes, Your= 4, Score = 62
 Yes, Your= 128, Score = 70
 Yes, Your= 32, Score = 78
 Yes, Your= 4096, Score = 86
 Yes, Your= 2, Score = 94
 Yes, Your= 1, Score = 100
End of Permutations Combinations, Summer 2014
```

The following activities occurring during the exam will be considered scholastic dishonesty:

1) running any program from the PC other than **Keil uVision**, or a calculator,
2) using material/equipment other than a pen/pencil.

Your signature is your promise that you have not cheated and will not cheat on this exam, nor will you help others to cheat on this exam:

Signed: ______________________________________ July 24, 2014

Procedure

First, you will log onto the computer and download the **zip** file from the website:
http://users.ece.utexas.edu/~ryerraballi/sa21014/Exam2

UNZIP the folder placing it **ON THE DESKTOP**. Within **Keil uVision** open the project, put your name on the first comment line of the file **Exam2.c**. Before writing any code, please build and run the system. You should get output like the figure above (but a much lower score). You may wish create backup versions of your program. If you wish to roll back to a previous version, simply open one of the backup versions.

My main program will call your subroutines multiple times, and will give your solution a performance score of 0 to 100. *You should not modify my main program or my example data.* Each time you add a block of code, you should run my main program, which will output the results to the `UART#1` window. After you are finished, raise your hand and wait for a TA. The TA will direct you on how to complete the submission formalities. The TA will run your program in front of you and record your performance score on your exam cover sheet. The scoring page will not be returned to you.

The exam has three parts a) through c), details of which are given in the starter code (Exam2.c):

Part a) The first subroutine you will write, called **`Permute`** computes and returns the number of ways **`N`** objects can be *arranged* in **`K`** slots (Permutations). One can compute **`Permute(N,K)`** as:

`N*(N-1)*(N-2)*… *(N-K+1)`.

For example:

```
Permute(5,2) = 5*4 = 20
Permute(4,4) = 4*3*2*1 = 24
Permute(3,1) = 3
```

Another way to think about the computation is there are **`K`** terms involved in computing **`Permute(N,K)`**. These **`K`** terms start at **`N`** and count **`K`** terms down.
Note exceptions: (i) If **`N`** or **`K`** is *zero* then result is 1; (ii) If **`N<K`** then the result is undefined so you must return -1.

Part b) The second subroutine you will write, called **`NChooseK`** computes and returns the number of ways **`k`** objects can be *choosen* out of **`N`** (Combinations). One can compute **`NChooseK`** as :

`(N*(N-1)*(N-2)*… *(N-K+1))/(K*(K-1)*(K-2)…*1)`

You will notice the numerator is simply **`Permute(N,K)`**, and the denominator can also be calculated by calling the Permute function with appropriate inputs. Again, note that the same exceptions that apply to Permutations also apply to Combinations.

Part c) The third subroutine you will write (**`BinomialCoeeficients`**) computes multiple Combinations which are the coefficients of the *binomial expansion*. For a given input **`N`**, there are (**`N+1`**) binomial coefficients: **`NChooseK(N,0), NChooseK(N,1), NChooseK(2),… , NChooseK(N).`**

Your routine will do two tasks:

i) compute the coefficients and put them in the array passed as the second input.
ii) return the sum of the coefficients.

Incidentally the sum of the coefficients always add up to **`2^N`**.

Important Notes:

- Your subroutines should work for all cases shown in the starter file.
- Handle the simple cases first and the special cases last.

- ***Calling your subroutines from within other subroutines will make your task easier***