CS 158/159 Lab 11

In this lab session you will complete the following tasks in the specified order. **Do not begin this lab until your lab sessions starts**. Individuals failing to remain on task during the lab will be assigned a zero for the lab, lab quiz, and will be considered absent for the lab. Web browsing, personal e-mail, social networking, and/or text messaging is not permitted during lab.

1. Record Attendance

Log on to your UNIX account on guru on a physical PC in the lab room (not on a wireless device) and enter the command **`attend`** to officially record your lab attendance. A confirmation e-mail will then be sent to you to verify that your attendance was correctly recorded. If you are late, you should still run the `attend` command and continue to participate in the remaining portion of the lab, but you will not be eligible to receive credit for the lab programming assignment. If you are present in the lab, but fail to run the `attend` command as expected, you will be marked as absent and likewise will not be eligible to receive credit for the lab programming assignment.

2. Review Material

The first task is to improve your understanding of the material from the book and the course programming standards (available in the course packet and in Blackboard) by working through a series of problems with the assigned lab partners of your group. You must show your lab instructor that you have successfully completed these tasks BEFORE you leave lab today. You will not receive points for the programming assignment, unless this task has been completed to the satisfaction of your lab instructor.

Solve the following problems related to material from Chapter 8:

Statement	**True / False**
An array is a collection of elements of the same data type.	
Placing the subscript of an element inside of square brackets is known as indexing.	
In a fixed-length array the size of the array is known when the program is written.	
Arrays must be declared and defined before they can be used.	
Array declarations will determine the type, name, and size of the array.	
For a value to be potentially used as an index it must be an integral value or an expression that evaluates to such.	
The name of an array is a reference to the address of where it begins inside the memory of the computer.	
The index value represents an offset from the beginning of the array to the element being referenced.	
Declaration and definition of an array will include a default initialization of all elements.	
If the number of values provided for initialization of an array is fewer than the size of the array then the remaining elements have no known value.	

The address operator is not necessary in a `scanf` to accept input for an individual array element when using the indexing technique.	
When accessing an array element the C language does not check whether the index is within the boundary of an array.	
Arrays can be passed in two ways; by individual elements or the whole array.	
Elements of an array, themselves individual values of a given data type, are passed by value from calling to called function.	
The called function cannot tell whether the value it receives comes from an array, an individual variable, or an expression that evaluates to the expected type.	
Individual elements of an array can be passed by address through the use of the address operator.	
The reason that the C language does not pass whole arrays by value is the extra stress it would put on the memory of the computer to make a copy of an array.	
The name of an array is a primary expression whose value is the address of the first element in the array.	
Indexed references to individual elements of an array are simply calculated addresses where the index value is added to the address represented by the name of the array.	
Passing the array name to a function allows changes in the called function to be available back in the calling function after it terminates.	
When passing a whole array to the function the total size of the array is necessary in the definition of the called function.	
It is only the starting point of the array in memory that is represented by the name of an array and not the ending point.	

What are the values in the array after the code below has been executed?

```
int x[5];
int i;

for (i = 0; i < 4; i++)
 x[i] = 2 * i - 1;
```

What are the values in the array after the code below has been executed?

```
int x[5] = {1, 3, 7};
int i;

for (i = 0; i < 4; i++)
 x[i + 1] += x[i];
```

3. Programming Assignment

The second task is to develop a program as a group which solves the given problem. This assignment is worth 15 points and will be **due 30 minutes prior to the start of your next lab session**. All assignment deadlines are firm and the ability to submit your assignment will be disabled after your deadline elapses. No late work will be accepted!

As you develop the program, you should rotate through the following roles approximately every 30 minutes. Do not allow the same person be the driver the entire time. It is not acceptable to designate a single individual to complete the assignment. Every individual group member should have a full understanding of all work submitted. Assignments are an opportunity to develop and demonstrate your understanding of course material.

Role	**Description**
Driver	The driver is in charge of the computer which includes entering code, saving, testing, and submitting. This individual should be soliciting the other two members for advice.
Navigator	The navigator's role is to look over the shoulder of the driver for syntactical errors, logical errors, and concerns related to course standards. The most common mistakes include failing to pair quotes, misplaced semi-colons, and improper placement of parentheses.
Manager	The manager may not be as close to the driver as the navigator but still plays an important role ensuring that the algorithm to be implemented is correct and can be tested using a variety of input to verify correctness. The manager is responsible for communicating to the teaching assistant who will be making the group's final lab submission.

This programming assignment does not have a single solution, and each group should collaborate together to develop their own unique solution. Submissions may be processed with comparison software and results will be used to detect unacceptable collaboration between groups. The development of your algorithm and the resulting code should only be discussed among your group members and course staff.

Your program must adhere to the course programming standards (available in the course packet and in Blackboard). Please review this document before submitting your assignment, because failure to adhere to it will result in a reduction in points. You program must include the designated program header (~cs15x/student/hdrProg) at the top of the program (which can be inserted in `vi` using the `hp` shortcut while in command mode). The header must include an appropriate description of your program and must contain the official Purdue career account e-mail addresses of each **contributing** group member. Do not include the e-mail address of anyone who did not actively participate in the program development. Failing to participate in the process to the satisfaction of all partners will result in a zero. Also note that course standards prohibit the use of advanced programming concepts not yet introduced in the course, unless otherwise specified.

Each of the example executions provided below represents a single execution of the program. Your program must accept input and produce output **exactly** as demonstrated in the example executions. Your program will be tested with the data seen in the examples below and an unknown number of additional tests making use of reasonable data. Do not include any example outputs with your submission.

A single program file (with the `.c` extension) must be submitted electronically via the `guru` server. An example submission was conducted during the first week in lab00. If you have a concern regarding how to submit work, please visit course staff prior to the assignment deadline.

Problem: Given the number base of a ten integer data set, identify for each integer that conversion from base 5 to 10 which contains the most 4 digits. Display the largest base when multiple conversions contain the same largest number of 4 digits. Do not display any value for which none of the conversions contain at least a single 4 digit. Input must be validated to ensure that the integer entered does not include an invalid digit for the given number base of the data set.

Example Execution #1:

```
Enter the base of the data set -> 10
Enter value #1 -> 1234
Enter value #2 -> 2341
Enter value #3 -> 3412
Enter value #4 -> 4123
Enter value #5 -> 2345
Enter value #6 -> 3456
Enter value #7 -> 4567
Enter value #8 -> 5678
Enter value #9 -> 6789
Enter value #10 -> 7890

Conversions with most fours:
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
###      Number      Base      Fours
 1:        14414        5          3
 2:         4445        8          3
 3:        23444        6          3
 4:       112443        5          2
 5:         4451        8          2
 6:         3456       10          1
 7:         4567       10          1
 8:        42142        6          2
 9:       204124        5          2
```

Note on Example Execution #1:

The tenth value (7890) does not have a conversion which contains a 4 digit.

Base 5 - 223030
Base 6 - 100310
Base 7 - 32001
Base 8 - 17322
Base 9 - 11736

Example Execution #2:

```
Enter the base of the data set -> 2
Enter value #1 -> 101
Enter value #2 -> 100
Enter value #3 -> 1110
Enter value #4 -> 1010100
Enter value #5 -> 110010100
Enter value #6 -> 100100100101
Enter value #7 -> 10000000
Enter value #8 -> 10000001
Enter value #9 -> 10000010
Enter value #10 -> 10000011

Conversions with most fours:
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
###      Number       Base      Fours
 2:           4        10         1
 3:          14        10         1
 4:          84        10         1
 5:         404        10         2
 6:        4445         8         3
 7:         242         7         1
 8:         243         7         1
 9:         244         7         2
10:         245         7         1
```

Example Execution #3:

```
Enter the base of the data set -> 3
Enter value #1 -> 200212
Enter value #2 -> 21012101
Enter value #3 -> 22112
Enter value #4 -> 313101

Error! Invalid digit for base 3

Enter value #4 -> 1121220100
Enter value #5 -> 220011
Enter value #6 -> 10010
Enter value #7 -> 200121
Enter value #8 -> 20102121
Enter value #9 -> 22110202
Enter value #10 -> 1110111011111

Conversions with most fours:
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
###      Number       Base      Fours
 1:        4014         5         2
 2:       40144         6         3
 3:         446         7         2
 4:       76404         8         2
 5:         804         9         1
 6:          84        10         1
 7:        2154         6         1
 8:       16444         7         3
 9:       14040         8         2
10:     1414144         9         4
```

Example Execution #4:

```
Enter the base of the data set -> 6
Enter value #1 -> 1050231
Enter value #2 -> 201052
Enter value #3 -> 552352
Enter value #4 -> 2051115
Enter value #5 -> 10233051
Enter value #6 -> 10312021
Enter value #7 -> 12525551
Enter value #8 -> 152532535
Enter value #9 -> 434415
Enter value #10 -> 250315

Conversions with most fours:
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
###      Number      Base     Fours
 1:       147753       8        1
 2:        64031       7        1
 3:        45932      10        1
 4:       564464       7        3
 5:       506534       9        1
 6:       514344       9        3
 7:      1454413       8        3
 8:      6041475       9        2
 9:       434415       6        3
10:        33342       9        1
```

Example Execution #5:

```
Enter the base of the data set -> 4
Enter value #1 -> 2022
Enter value #2 -> 2133
Enter value #3 -> 21302
Enter value #4 -> 21333
Enter value #5 -> 30110
Enter value #6 -> 31001
Enter value #7 -> 122303
Enter value #8 -> 132221
Enter value #9 -> 132331
Enter value #10 -> 133202

Conversions with most fours:
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
###      Number      Base     Fours
 2:          423       6        1
 4:         2543       6        1
 5:         1424       8        2
 9:         2641       9        1
10:         3742       8        1
```

Example Execution #6 (no value in the set has an equivalent with a 4 digit):

```
Enter the base of the data set -> 10
Enter value #1 -> 755
Enter value #2 -> 1275
Enter value #3 -> 2000
Enter value #4 -> 15767
Enter value #5 -> 22136
Enter value #6 -> 100191
Enter value #7 -> 390810
Enter value #8 -> 53317
Enter value #9 -> 651
Enter value #10 -> 165707

Conversions with most fours:
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
###      Number      Base     Fours
```

Example Execution #7 (validation requirements demonstrated):

```
Enter the base of the data set -> 11

Error: Please enter a base in the range [2, 10]

Enter the base of the data set -> 9
Enter value #1 -> 505202
Enter value #2 -> 834651
Enter value #3 -> 120800
Enter value #4 -> 135724
Enter value #5 -> 212313
Enter value #6 -> 415514
Enter value #7 -> 777777
Enter value #8 -> 444444
Enter value #9 -> 404044
Enter value #10 -> 913476

Error! Invalid digit for base 9

Enter value #10 -> 134769

Error! Invalid digit for base 9

Enter value #10 -> 476993

Error! Invalid digit for base 9

Enter value #10 -> 404400

Conversions with most fours:
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
###      Number      Base     Fours
 1:     34032204       5        2
 2:      4132450       7        2
 3:      4312234       5        2
 4:      1440034       6        3
 5:       366644       8        2
 6:       742044       8        3
 7:     13544454       6        4
 8:       444444       9        6
 9:       404044       9        4
10:       404400       9        3
```

Additional Notes:

- The data set will always include 10 values after validation. Validation depends on the base entered. See the seventh test case for an example of validation expectations.
- All arrays must be of fixed-length using a symbolic/defined constant to declare the size.
- Course standards prohibit the use of programming concepts not yet introduced in lecture. For this assignment, you can only consider material in the first 8 chapters of the book, notes, and lectures. Use of advanced programming constructs beyond this material would result in a loss of points.

4. Group Coordination

The next task is to collaborate as a group to determine who will submit your program assignment for grading. **Only one person per group will make submissions for the entire group**. Lab instructors are not required to grade submissions from multiple members of the same group to determine which submission you actually want graded. Also, the group member should not always be making the submission each week. Record the names and official Purdue career account e-mail addresses of all three lab partners here and put a checkmark in the Submitter column for the person responsible for making the submission. Your group should turn in this page of information to their lab instructor before you leave lab today.

Name	Purdue career account e-mail address	Submitter

If possible, it is a good idea to submit your work for grading prior to leaving lab today, even if it is not completely finished. This will allow each lab partner to **verify their contact information in the assignment header is correct** as each would then receive an e-mail verifying the submission. You may make multiple submissions before the deadline, but only the last attempt is retained and graded. Any previous submissions will be over-written and cannot be recovered. The submission script will reject the submission of any file that does not compile. A program must compile to be considered for partial credit.

If your group is unable to complete your assignment during the lab session, then it is expected that your group will meet outside of class to finish and submit the programming assignment. That is why you should exchange contact information during lab today! Before you leave, you should discuss when and where the group will meet next and when you plan for final submission to be made. If a group member you have entrusted to do the submission cannot be contacted and he/she is the only one who has a copy of the program, then the rest of the group would have to essentially start over in order to complete the programming assignment. Thus, it is a good idea for each person to have a copy of the program.

Consider the members of your group to be resources to assist you as you learn the material in this course. As a group, you may want to arrange a time to visit the TA office hours together or attend a particular Supplemental Instruction session together.

5. Lab Quiz

The final task will be to take the lab quiz which be made available in Blackboard during the last part of your lab today. **Lab quizzes are individual efforts and you may not use any resources while completing the quiz.** The quiz is worth 5 points and you must be present in lab session in order to credit for the quiz. All problems on lab quizzes will be multiple-choice or true-false. The quiz will emphasize material from book and the appropriate course programming standards. You will be given 10 minutes to complete the quiz and questions will be presented one at a time and cannot be revisited. Be sure to save your answers to each question and to finish your quiz to ensure it is submitted for grading. Any quiz that is taken when it is not being proctored by the lab instructor will receive zero points.