CS 158/159 Lab 12

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**
The memory address represented by the name of an array is added to the index value specified to determine where in memory the desired element can be found.	
All elements of one array can be assigned to another through the use of the **assignment operator** and the name of each array (example: `x = y`).	
While the default technique of **passing array elements is by value** it is possible to **pass elements by address** using the `&` operator (and the `*` operator in the function being called).	
If more than one element of an array is passed to a function in a single function call then those elements are **passed by address**.	
Using the name of an array in the data list of a single `printf` function will result in the **output of all elements** of the array.	
All arrays sent to a given function must be of the same defined size.	
The **selection sorting** algorithm will complete at most one exchange involving two elements per pass.	
The **selection sorting** algorithm can only be used to sort data in an ascending order (from smallest to largest).	

On the final pass through the **selection sorting** algorithm TWO values are brought over from the unsorted list into the sorted list.	
The **bubble** sorting algorithm compares neighboring elements in the unsorted list of the array and swaps their positions when they are not in the desired order.	
The **bubble** sorting algorithm is optimized to stop the process when the array is detected as being sorted.	
Once the **insertion sort** places a value in the sorted list that value will never move again in the remainder of the passes.	
The **insertion** sorting algorithm begins with one value in the sorted list before the first pass.	
To sort an array of `SIZE` elements a total of `SIZE - 1` **passes** are required to guarantee that the data is sorted using the selection, bubble, or insertion sort.	
When an array is not sorted and the data in the array is unique the **linear** searching algorithm requires every element to be compared with the target before it can be concluded that the target is not present in the array.	
The **binary** searching algorithm will always find a target in an array faster than the linear searching algorithm.	
The **binary** searching algorithm will terminate when the `first` variable is greater than the `last`.	
The **linear** and **binary** searching algorithms perform in the same amount of time that a given value is not present in an array.	
The **binary** searching algorithm should always be used for searching.	

What is the content of the array after the function call has been made?

```
#define SIZE 8

void changeArray(int[]);

int main()
{
 int x[SIZE] = {2, 8, 1, 7, 3, 6, 4, 5};
 changeArray(x);
 return 0;
}

void changeArray(int x[])
{
 int i;

 for(i = 0; i < SIZE / 2; i++)
 {
  x[i] -= x[SIZE - i - 1];
  x[SIZE - i - 1] += x[i];
  x[i] = x[SIZE - i - 1] - x[i];
 }
}
```

Solve problem #19 on page 543:

Solve problem #20 on page 543:

Solve problem #21 on page 543:

Solve problem #25 on page 544:

Solve problem #26 on page 544:

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: Modify your lab #11 effort to display all number system (base) conversions of the values in the data set that have more four digits than the original number. It is possible that a given value may have more than one conversion that meets this definition. Your program from lab #9 may help to verify such cases. In the example execution below each of the conversions (and base in parentheses) are identified. The resulting output should sort by number (largest to smallest) and then by value within each base (smallest to largest).

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 more fours:
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
###      Number      Base     Fours
 1:       204124         5         2
 2:       140203         5         1
 3:       112443         5         2
 4:        42142         6         2
 5:        23444         6         3
 6:        14414         5         3
 7:         5414         6         2
 8:         4451         8         2
 9:         4445         8         3
```

Example Execution #2:

```
Enter the base of the data set -> 8
Enter value #1 -> 137
Enter value #2 -> 2115
Enter value #3 -> 3230
Enter value #4 -> 672
Enter value #5 -> 715
Enter value #6 -> 1111
Enter value #7 -> 1373
Enter value #8 -> 1531
Enter value #9 -> 1700
Enter value #10 -> 2120

Conversions with more fours:
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
###      Number      Base     Fours
 1:        13404         5         2
 2:        13401         5         1
 3:        11452         6         1
 4:        11412         5         1
 5:         5040         6         1
 6:         4631         7         1
 7:         4320         5         1
 8:         4240         6         2
 9:         3545         6         1
10:         2541         7         1
11:         2413         6         1
12:         2140         7         1
13:         2045         6         1
14:         2014         6         1
15:         1464         7         2
16:         1456         9         1
17:         1453         9         1
18:         1104        10         1
19:          541         9         1
20:          461        10         1
21:          442        10         2
22:          340         5         1
23:          164         7         1
```

Example Execution #3 (validation of base):

```
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 -> 0

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

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 more fours:
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
###      Number      Base     Fours
 1:        14501       6        1
 2:         4445       8        3
 3:         3104       5        1
 4:         2341      10        1
 5:         1004       5        1
 6:          624       8        1
 7:          488       9        1
 8:          404      10        2
 9:          334       6        1
10:          314       5        1
11:          245       7        1
12:          244       7        2
13:          243       7        1
14:          242       7        1
15:          154       9        1
16:          124       8        1
17:           84      10        1
18:           24       5        1
19:           14      10        1
20:            4       5        1
21:            4       6        1
22:            4       7        1
23:            4       8        1
24:            4       9        1
25:            4      10        1
```

Example Execution #4 (validation of data set):

```
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 more fours:
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
###      Number      Base     Fours
 1:    111324043       5        2
 2:    104340020       5        2
 3:     34032204       5        2
 4:     30344240       5        3
 5:     14342031       6        2
 6:     13544454       6        4
 7:     13020442       5        2
 8:     10224302       6        1
 9:      5142404       6        3
10:      4312234       5        2
11:      4132450       7        2
12:      3644500       7        2
13:      2413020       6        1
14:      1614162       8        1
15:      1440034       6        3
16:      1321043       6        1
17:      1034301       7        1
18:       742044       8        3
19:       465010      10        1
20:       422205       7        1
21:       366644       8        2
22:       299054      10        1
```

Example Execution #5:

```
Enter the base of the data set -> 10
Enter value #1 -> 14
Enter value #2 -> 9
Enter value #3 -> 22
Enter value #4 -> 10
Enter value #5 -> 16
Enter value #6 -> 13
Enter value #7 -> 11
Enter value #8 -> 18
Enter value #9 -> 12
Enter value #10 -> 20

Conversions with more fours:
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
###      Number      Base     Fours
 1:           42         5         1
 2:           40         5         1
 3:           34         6         1
 4:           24         6         1
 5:           24         7         1
 6:           24         8         1
 7:           24         9         1
 8:           14         5         1
 9:           14         6         1
10:           14         7         1
11:           14         8         1
12:           14         9         1
```

Example Execution #6 (no new line in final message):

```
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

There are no conversions present with more four digits.
```

Additional Notes:

- The data set will always include 10 values after validation. Validation depends on the base entered. See the fourth test case for an example of this expectation.
- The base must be validated to be 2 – 9. See the third test case for an example of this expectation.
- The resulting data set of conversions will never exceed 50.
- 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.