Lab 04 Specification – Examine Internal Data Representation
Due (via your git repo) no later than 8 a.m., Friday, 28 September 2018.
50 points

Lab Goals

- Practice truth table
- Practice Number Conversion Rules

You are required to work in groups of two in this lab. So find a team member to work with.
May be you could reconnect to your team mate, with whom you partnered for the in-class exercises.
If you needed a team member, ask in Slack. Let us connect to each other and learn from each other!
Make sure to fill out the google form (link provided to your allegheny email account) by tomorrow morning 8.00 am, with your team details.
The Submission GitHub account is required to be filled out, which is one of your team members Git account. The repository associated with the submission Git account provided, would be used for grading your team's work.
Once grading is done, the graded report shall be uploaded to the submission Git repository provided in google form and all team members will receive same grade for the lab.

Suggestions for Success

- Take a look at the suggestions for successfully completing the lab assignment, which is available at: `https://www.cs.allegheny.edu/sites/amohan/resources/suggestions.pdf`

Learning Assignment

To do well on this assignment, you should also read

- Class discussion notes and slides.

Assignment Details

In this lab assignment, you will work on automating generation of a truth table for displaying binary form of different data (decimal values). Next you will also work on writing programs to automate conversion of data from one format to another.

Part 1: Truth Table Generator(25 points)

Write a program called ttGen.c, that automate the truth table for any random user defined ”n” value. We had an entire class discussion on Truth Tables and looked at the technique to generate a truth table for the first 16 integers. Now let us suppose, we would like to do it for a larger number. And more interestingly, we would like to write a program that automates this process. Ah, does that sound interesting! Okay, let us get started to have some fun. See the technical requirements for implementing this part below:

1. Your program should have a main function, that should print the following message as a starting point:

   ”Hey! Welcome to the truth table generator. Please enter the number of rows:”

2. Based on the user provided number of rows (also known as the ”n” value), the total number of bits should be calculated.

   So, we know that to generate a truth table with 4 rows, we need 2 bits.

   To generate a truth table with 8 rows, we need 3 bits.

   To generate a truth table with 16 rows, we need 4 bits.

   To generate a truth table with 32 rows, we need 5 bits.

   How about generating a truth table with 25 rows? we still need 5 bits. So how to compute the number of bits, given the number of rows?

   ```
   Let X be the total number of bits to generate "n" rows;
   If (n is power of 2) Then
           $2^X$ = n;
           Find X;
   Else
           Find a number Y such that:
           [Y > n; Y is a power of 2; Y is closest to n;]
           $2^X$ = Y;
           Find X;
   End If
   ```

   Hint: Well, do you recall the log and a ceil builtin functions, that are part of the ”math.h” library? May be you should utilize those functions to implement the algorithm outlined above.

3. The next step is to use a correct data structure to hold the truth table data. How about using a multi-dimmensional array?

   Here is an example of initializing, assigning values, and printing a multi-dimmensional array in C.

   ```
   #include <stdio.h>
   #include <math.h>
   int main(){

           int multiarr[rows][cols];
           /* filling the array elements with a value*/
           for (int i = 0; i < rows; i++){
                   for (int j = o; j < cols; j++){
                           multiarr[i][j] = i + j;
                   }
           }

           /* printing the array elements*/
           for (int i = 0; i < rows; i++){
                   for (int j = o; j < cols; j++){
                           printf("%d\t",multiarr[i][j]);
                   }
   ```

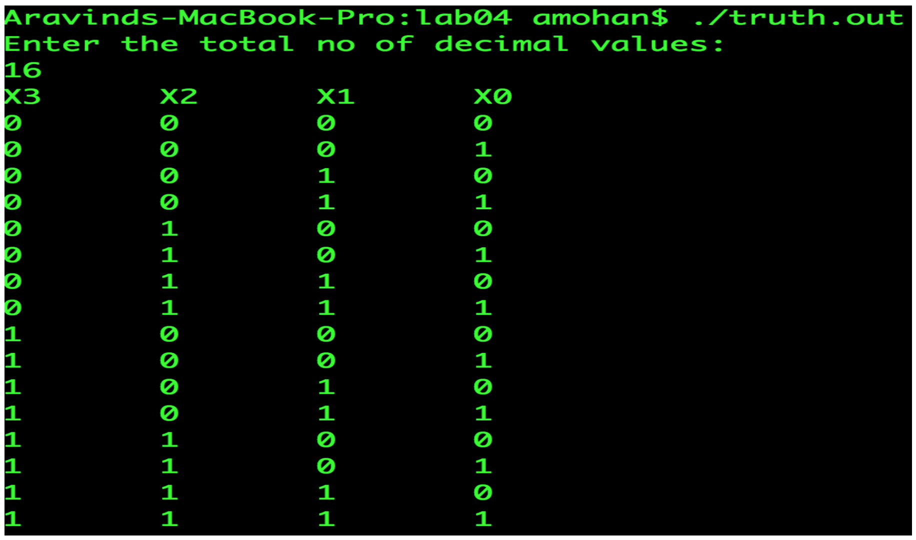
```
                    printf("\n");
            }

    }
```

Recall what is the total number of rows and cols for the multi-dimmensional array from last class discussion?

An important question is how to assign values to each cell in the array? We discussed about this during our class discussions. It is a good exercise to brainstorm with your team mate and think about how to define a generalized formal approach for doing this in your program. The actual formula is hidden from you, so as to make this a more interesting programming exercise. I recommend not to look at the template shown in the last page of this specification, but there is some hint given in the last page. If you had tried enough and want the formula, I suggest sending a private Slack message to me regarding this. I will provide the formula through my response. But, I recommend not to share these as such to other class members, so as to keep this interesting. If someone needs the formula, they should ask me first. There might be some teams, who would like to keep trying and figure out the formula themselves. I don't want the formula to leak in the air and then it is no more interesting for all the teams! But, I could tell that this is an interesting formulization in CS and a very good thinking exercise to go through.

I could reveal just a little bit for now, "inverse of exponential function is a logarithm function in math!".

4. Finally, the truth table need to be displayed in the console. A copy of the expected result (with no of rows = 16) is shown below for your reference!



```
Aravinds-MacBook-Pro:lab04 amohan$ ./truth.out
Enter the total no of decimal values:
16
X3      X2      X1      X0
0       0       0       0
0       0       0       1
0       0       1       0
0       0       1       1
0       1       0       0
0       1       0       1
0       1       1       0
0       1       1       1
1       0       0       0
1       0       0       1
1       0       1       0
1       0       1       1
1       1       0       0
1       1       0       1
1       1       1       0
1       1       1       1
```

Part 2: Basic Number Converter (25 points)

Write a program called converter.c, that does number conversion from one format to another. There are two conversion schemes included in this part. First is to convert a given decimal to binary and second is to convert a given binary to decimal. See the technical requirements for implementing this part below:

1. Your program should have a main function, that should print the following message as a starting point:

   "Hey! Welcome to the number converter. Please enter 1 for binary to decimal; 2 for decimal to binary conversion:"

2. Based on the user provided input, either the "binaryToDecimal" function or the "decimalToBinary" function should be triggered.

3. The rules of these conversion schemes and the implementation details of ”binaryToDecimal” and ”decimalToBinary” are provided as follows:

   Converting Binary to Decimal

   - Starting with the most significant bit (left to right), repeatedly multiply by 2, adding each bit as we move along.
   - For example, 1010111_2

     - $(0+1)\times 2 = 2$
     - $(2+0)\times 2 = 4$
     - $(4+1)\times 2 = 10$
     - $(10+0)\times 2 = 20$
     - $(20+1)\times 2 = 42$
     - $(42+1)\times 2 = 86$
     - $(86+1) = 87$

     - Solution is 87_{10}

   Converting Decimal to Binary

   - Divide repeatedly by 2 and retain the remainders. Continue until the quotient = 0.
   - For example, 245_{10}

     - $245 \div 2 = 122$ R = 1 Least Significant Bit
     - $122 \div 2 = 61$ R = 0
     - $61 \div 2 = 30$ R = 1
     - $30 \div 2 = 15$ R = 0
     - $15 \div 2 = 7$ R = 1
     - $7 \div 2 = 3$ R = 1
     - $3 \div 2 = 1$ R = 1
     - $1 \div 2 = 0$ R = 1 Most Significant Bit
     - Solution is 11110101_2

Part 3: Advanced Number Converter (EXTRA CREDIT)

Write a program called advanced.c, that does number conversion from one format to another. There are two conversion schemes included in this part. First is to convert a given hexadecimal to binary and second is to convert a given binary to hexadecimal. See the technical requirements for implementing this part below:

1. Your program should have a main function, that should print the following message as a starting point:

   ”Hey! Welcome to the advanced number converter. Please enter 1 for binary to hexadecimal; 2 for hexadecimal to binary conversion:”

2. Based on the user provided input, either the ”binaryToHEX” function or the ”hexToBinary” function would be triggered.

3. In order to fully implement this part, you would need access to the truth table data structure and reference the hexadecimal (A-F) equivalent for decimal values.

4. The rules of these conversion schemes and the implementation details of ”binaryToHEX” and ”hexToBinary” are provided as follows:

Converting Binary to Hexadecimal

- Four binary digits range from 0000 = 0 to 1111 = 15. This is the same as one hexadecimal digit. So, group sets of 4 binary digits and convert each to one hexadecimal digit:
  - 110 0101 1101_2
  - 6 5 D_{16}
  - Solution: $11001011101_2 = 65D_{16}$

Converting Hexadecimal to Binary

- Expand each hexadecimal digit into the corresponding 4 binary digits:
  - 1 2 3 4 A F 0 C
  - 0001 0010 0011 0100 1010 1111 0000 1100
  - Solution: $1234AF0C_{16} = 00010010001101001010111100001100_2$

Submission Details

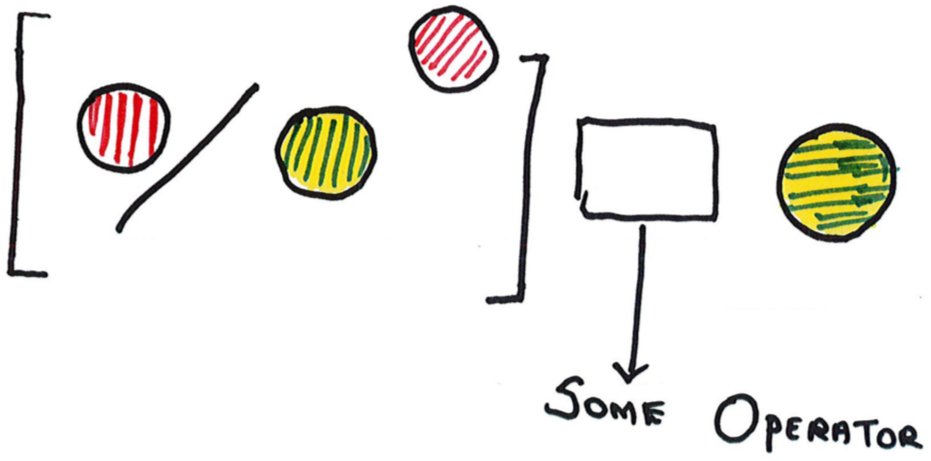
1. Your lab assignment solution file must be uploaded to your repository by the due date and time.
2. Questions about the lab? Bring them to class on Tuesday morning!

Before you turn in this assignment, you must ensure that the course instructor has read access to your git repository that is named according to the convention `cs200F2018-<your user name>`.

Grading Rubric

1. If you complete Part 1 fully, as per the requirement outlined above, you will receive a total of 25 points.
2. If you complete Part 2 fully, as per the requirement outlined above, you will receive a total of 25 points.
3. If you complete Part 3 fully, as per the requirement outlined above, you will receive an additional (1 point) towards your final class score. For example, if your final class score is ”89/100”, by completing the extra credit successfully you will receive ”90/100”. It is my plan to provide a total of 5 extra credit opportunities either through the lab assignments or through additional questions in the quizzes or exams. Again, please be advised that I am committed towards your success in this course. If you like, dislike this approach, take a minute to vote your opinion by clicking the poll link below:

   `https://pollev.com/drmohan958?_ga=2.73397395.1858074201.1537492875-1183467583.1537492875`
4. If you fail to fill out the google form by tomorrow 8.00 am, your work will not be graded and there will be no points awarded for your submission towards this lab assignment.
5. Failure to upload the lab assignment code to your git repo, will lead you to receive ”0” points given for the lab submission.
6. There will be a partial credit (30% of the points allocated for the part) given if your code in (Part 1, 2) compiles correctly but not able to produce the expected result. Note: there will be no partial credit awarded for the extra credit question.
7. There will be no partial credit awarded if your code doesn’t compile correctly. It is highly recommended to validate if the correct version of the code is being submitted before due date and make sure to follow the honor code policy described in the syllabus.
8. If you needed any clarification on your lab grade, talk to the instructor. The lab grade may be changed if deemed appropriate.


Some Operator

Wish you
all the best