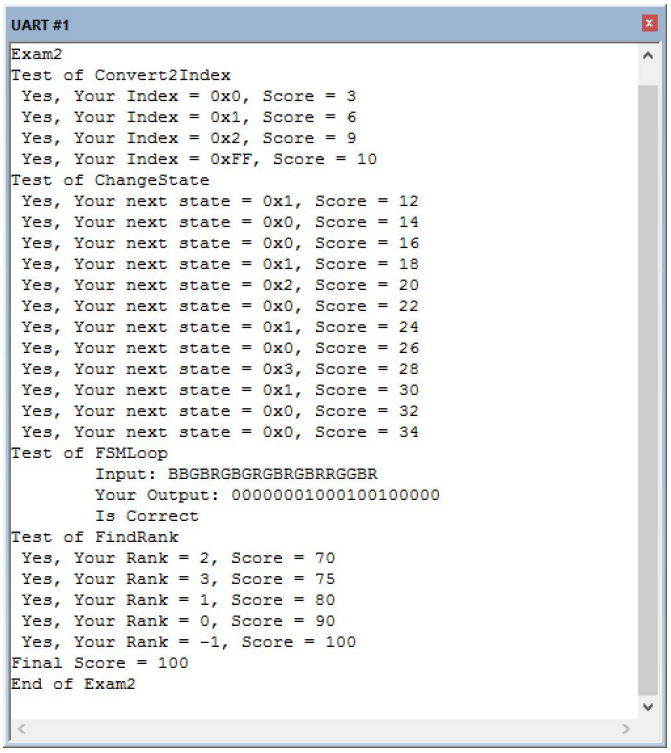
First:_____ Last:_____ Instructor (Circle): **NT MT ME RY**

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		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 75 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). 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 access your network drive or the internet. You are not allowed to discuss this exam with other EE319K students until Friday evening.



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) communicating with **anyone else** except for the instructors **by any means** about this exam until Friday
3) using material/equipment other than a pen/pencil,
4) hard-coding so it outputs answers that give points without actually solving the problem,
5) modifying anything other than **Exam2CPart.c and Exam2AsmPart.c**

Students caught cheating will be turned to the Dean of Students.

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: _____ April 6, 2016

Procedure

First, you will log onto the computer and download files from the web as instructed by the TAs.

Web site: **http://users.ece.utexas.edu/~valvano/Volume1/Exam2ETTY**
User: etty
Password: 222

UNZIP the folder placing it **ON THE DESKTOP**. You are not allowed to archive this exam. Within **Keil uVision** open the project, put your name on the first comment line of the file **Exam2CPart.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 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 functions 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.

Important Notes:

- Your functions should work for all cases given to it by the grader.
- The description of functions here is less detailed than that in the comments above the function in the source files. Refer to them before attempting your solution.
- ***This exam is different from other exams. You are being asked to write two routines in assembly and debug one C function that calls your assembly subroutines, all pertaining to an FSM implementation; The other C function you are asked to write finds the rank of a student in a class.***

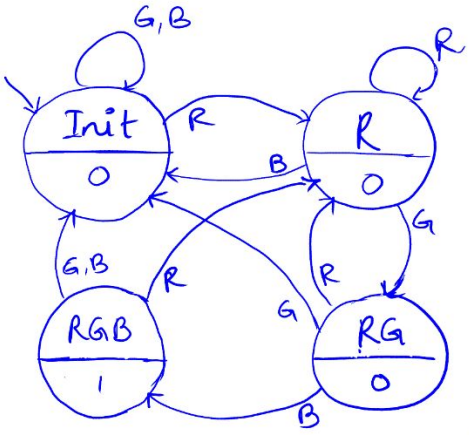
The exam has four parts a) through d), details of which are given in the starter code (**`Exam2CPart.c`** and **`Exam2AsmPart.s`**).

Parts a, b, and c, are related to a Moore Finite State machine that implements a detector. A certain tester for a TV screen flashes the LEDs in a random sequence of red (R), green (G), blue (B). When the sequence RGB is detected, an output 1 is produced. Otherwise, the output is 0.

Here's an Example:

Sequence: `BBGBRGBGRGBRGBRRGGBR`
Output: `00000001000100100000`

The FSM graph is given on the right. The implementation given to you has bugs in translation from Graph to Code and the FSMLoop engine. In addition your code calls routines that you are responsible for and routines that we provided.



Part a) Write an **assembly** subroutine called **`Convert2Index`** that converts the char input `R,G,B` to an index `0,1,2` respectively. Here is the function prototype:

```
uint32_t Convert2Index(char in);
```

<note> Fix FSM structure in C before doing part b </note>

Part b) Write an **assembly** function called **`ChangeState`** that changes the current state based on current state and input. Here is the function prototype in C for your reference:

```
void ChangeState (uint32_t *st, uint32_t in);
```

The first input (st) is address of the current state variable (unsigned 32-bit). The second input is the input value (unsigned 32-bit).
Note: This function uses *call-by-reference* for its first parameter and MUST be written in assembly.

Part c) Fix the **C implementation** of the FSM in **`Exam2CPart.c`.** Errors may be present in both the state machine declaration (**`STyp FSM`**) as well as the FSM engine (**`FSMLoop`**).

Part d) This is unrelated to the FSM portion of the exam. In this part, you have to write the function **FindRank** in C that finds the rank of a student in an array of student records:

```
int8_t FindRank( struct Rec_t students[], char name[]);
```

The first input to the function is **`students`**, which holds an array of student records. Each student has a name and score. The array is terminated by a student record with a score of -1. All other students have a score 0 <= score <= 100). No two students have the same score, so ranks are unique.
The second input is **`name`**, which holds a null-terminated array of chars (a String)

Your function must check if the given **`students`** array is sorted in ***ascending*** order of scores. If it is not sorted then return a -1 for student rank. If it is sorted then return the rank of the student whose **`name`** is passed. If student name is not found in the array of students then return a 0.
Notes:

The student with the best (highest) score has a rank of 1 and will be last entry if the array is sorted in ascending order.
To help you out, we are providing a **`StringMatch`** function that compares two null-terminated strings and returns a `1/0` based on whether they match or do not match respectively:

```
uint8_t StringMatch(char A[], char B[]);  //Given to you
```

Submission Guidelines:

- Log onto Canvas and submit your **Exam2CPart.c** and **Exam2AsmPart.s** source files into the Exam2 submission link. Be careful because only one submission will be allowed.