Coding Standard eYRC/eYRC-Plus 2014

- The following documentation and comment styles are to be used for the code submitted by the teams.
- Replace all the ***<Description>*** tags from the comments below to add appropriate content for your application.

1. File Level Comments

- Each user's code file should start with **File Level** comments in the format as follows:
  - /*
  - *
  - * Team Id: <Team Id>
  - * Author List: <Name of the team members who worked on this function
  - * (Comma separated eg. Name1, Name2)>
  - * Filename: <Filename>
  - * Theme: <Theme name -- Specific to eYRC>
  - * Functions: <Comma separated list of Functions defined in this file>
  - * Global Variables: <List of global variables defined in this file, None if no global
  - * variables>
  - *
  - */

2. Function Level Comments

- Each function should have the following comment section before it :
  - /*
  - *
  - * Function Name: <Function Name>
  - * Input: <Inputs (or Parameters) list with description if any>
  - * Output: <Return value with description if any>
  - * Logic: <Description of the function performed and the logic used
  - * in the function>
  - * Example Call: <Example of how to call this function>
  - *
  - */

3. Variable Comments

- In general the variable/function names should be descriptive enough to give a good idea of what the variable is used for.,For example, variable names like 'black_line_threshold_value', 'left_motor_turn_right' are preferrable and makes your code readable. Variable names like 'a', 'b' and 'temp' are not acceptable variable names.
- In some cases, variable names might require some description for which the following format can be used:
  - // Variable Name: Description of the variable and the range of expected values of the variable

4. Implementation Comments

- In your implementation/actual code, you should have comments in tricky, non-obvious, interesting, or important parts of the code.

- The comments can be of the format as below:
  - // Describe what the code below is doing

An Illustrative Example:
We provide sample comments in a rudimentary program that outputs the Fibonacci Series (For more information on Fibonacci Series visit: http://en.wikipedia.org/wiki/Fibonacci_number). Please note that this is not the complete and perfect example of generating Fibonacci Numbers but acts as a simple way to illustrate the coding style and comments explained above.

```
/*
 * Team Id: 10000
 * Author List: Amiraj Dhawan, e-Yantra Team
 * Filename: fibonacci.c
 * Theme: Cargo Sorting – eYRC Specific
 * Functions: print_fibonacci_series(int) , main()
 * Global Variables: NONE
 *
 */
#include<stdio.h>


/*
 * Function Name: print_fibonacci_series
 * Input:         num_elements -> integer which stores the number of elements of
 *                the fibonacci series to be printed
 * Output:        prints the first num_elements of the fibonacci series
 * Logic:         The next element of the Fibonacci series is given by
 *                next = current_element + prev_element
 *                The code loops for num_elements and prints out the next
 *                element
 * Example Call:  print_fibonacci_series(10);
 *
 */
void print_fibonacci_series(int num_elements){
      int first = 0, second = 1, next;
      printf("First %d terms of Fibonacci series are :-\n", num_elements);

      //counter: will iterate from 0 to (num_elements - 1)
      for ( int counter = 0 ; counter < num_elements ; counter++ ) {
            if ( counter <= 1 )
                  next = counter;
            else {
                     // The next element is equal to the sum of the current
                        element (second variable)and the previous element (first
                        variable)
                     next = first + second;
                     // first element becomes the second element and second
                        element becomes the next element for the next loop
                        iteration
                     first = second;
                     second = next;
            }
```

```
            printf("%d\n", next);
        }
}

/*
 * Function Name: main
 * Input:         None
 * Output:        int to inform the caller that the program exited correctly or
 *                incorrectly (C code standard)
 * Logic:         Ask the user to input the number of elements required from the
 *                Fibonacci Series and call the function  print_fibonacci_series
 * Example Call:  Called automatically by the Operating System
 *
 */
int main() {
        int num_elements;

        //    Ask the user to input the number of elements required
        printf("Enter the number of terms\n");

        scanf("%d",&num_elements);

        //    Call the function to print the first num_elements of the Fibonacci
              Series
        print_fibonacci_series(num_elements);

        return 0;
}
```

```
/*
 *      Following a coding style might look to be tedious at first but is one of the most important
 *      thing to be done while developing any piece of code. This ensures that it is readable so
 *      that others can understand what your code is doing. Even you yourself may find it useful
 *      after some time!
 */
```

"Any fool can write code that a computer can understand. Good programmers write code that humans can understand." - [Martin Fowler]

"Programs must be written for people to read, and only incidentally for machines to execute." - Hal Abelson & Gerald Jay Sussman

---

"Always code as if the guy who ends up maintaining your code will be a violent psychopath who knows where you live." - [Rick Osborne]