CS 341 LAB 2

All programs MUST be suitably commented at every line of code. Else you will lose 1/3rd of the points for that program.

1.BASIC INPUT/OUTPUT (1 Point)
Write an assembly code that prompts user to give their name. After receiving the name, the code should wish the user hello. Name the file **hello.s** .
Note: You can restrict the size of the name to 30 characters.

INPUT : <user name>
OUTPUT : "Hello <username>"

SAMPLE OUTPUT :
```
Please enter your name : Donald J. Trump
Hello Donald J. Trump
```

Hint :

a) Strings can be stored in Data Segment of a MIPS program (Use .word).
b) Read about "syscall"

2. ARRAY in MIPS (2.5 Points)
Create a new program having an array of size 8 as static data (arrays are stored in Data Segment of a MIPS program). You are supposed to use ".align" assembler directive for this(use google).
Now write a loop in main to read 8 values from console (use syscall) and store in the 8 array locations declared above. Calculate the sum of values at odd locations and even locations and store in different registers. Output a line stating which one is larger, followed by the larger value.
Name the file **array.s**

SAMPLE OUTPUT:
```
Enter 8 values : 1 2 3 4 5 6 7 8
Sum of values at even locations is larger
20
```

3. FUNCTIONS - BASICS (1.5 Points)
Create a new file **even.s** .
Write a function called ISEVEN which takes as input an integer. The output of the function is 1 if the integer is even else 0.
Call this function from main with the given input using the jal instruction. Once the function is complete store the output in appropriate register(s) and return to main using jr instruction.

The program should have proper prompts to take input from the console and write onto it.

SAMPLE OUTPUT:
Give a number : 5
5 is not even

4. FUNCTIONS - ITERATION (2 Points)
Create a new file **factorial.s**
Write a function called FACTORIAL which takes as input a positive integer and ***iteratively*** (NOT RECURSIVELY) calculates the value of factorial of that number. The output of the function is the factorial of the input number.
Call this function from main with the given input using the jal instruction. Once the function is complete store the output in appropriate register(s) and return to main using jr instruction.

Note: Assume that the numbers are small enough such that their factorials would fit in the registers

SAMPLE OUTPUT:
Give a number : 5
Factorial is : 120

5. STRUCTURES - BASICS(3 Points)
You are given a C++ code ratio.cpp shown below Translate it into assembly code. You may use the following snippet for declaring and accessing the global ratios using static memory allocation:

```
  .data
  .align 2
ratio1:  .word   0,0      #default initialisation
ratio2:  .word   0,0      #default initialisation

  .text
la $s0, ratio1   # Address of ratio1 is obtained in $s0 for future use
la $s1, ratio2   # Address of ratio2 is obtained in $s1 for future use
```

You are supposed to pass only two arguments to the function “add”. The final answer for the function “void add(ratio& r, ratio& s)” should be present in r (ratio1 in the assembly code snippet). Assume that the input will be non-zero integers.

ratio.cpp :

```
#include <iostream>
using namespace std;
typedef struct ratio{
  int numerator;
  int denominator;
} ratio;

ratio r1,r2;

void add(ratio & r, ratio & s) {
  r.numerator = r.numerator*s.denominator + s.numerator*r.denominator;
  r.denominator = r.denominator*s.denominator;
}

int main(){
  cin >> r1.numerator >> r1.denominator;
  cin >> r2.numerator >> r2.denominator;
  add(r1, r2);
  cout << r1.numerator << "/" << r1.denominator << endl;
}
// Code ends here
```

EXTRA CREDITS[1 points] :

1. Create a situation where not using “.align” directive creates error. Submit it in the form of a program that causes the error to occur suitably commented. Name the program error.s.