Nombre: _____
Código: _____

1. Preliminares del parcial

1.1. Estructura de directorios

En la página `http://www1.eafit.edu.co/fcardona/cursos/st0244/parcial03` se encuentra ficheros que son necesarios para configurar cygwin para el parcial.

1.1.1. *Script* de descarga automática

A las 6:00 pm. hacer lo siguiente en la terminal:

```
$ wget http://www1.eafit.edu.co/fcardona/cursos/st0244/st0244Parcial03.sh
$ chmod + st0244Parcial03.sh
$ ./st0244Parcial03.sh
```

Seguir las indicaciones que están en la pantalla.

1.2. Problemas del parcial

Son tres los problemas del parcial que se encuentra en las carpeta:

```
st0244
 + 244s$USERNAME
    + parciales
       + parcial03
          + helloworld
          + longlines
          + midtermexam
```

Las carpetas `longlines` y `midtermexam` son en C++, cada una de ellas tiene un `Makefile` que sirve para compilar y también para probar que los programas están correctos.

```
$ make run1
$ make run2
```

La carpeta `helloworld` está en Ruby. Ella tiene un fichero especial llamado `Rakefile`, este sirve para correr las pruebas:

```
$ rake run1
$ rake run2
```

Al ejecutar las pruebas si está pasan no aparece ningún mensaje de error.

En cada pregunta se indica el directorio donde esta el programa, el fichero que tiene que modificar (modifique únicamente ese fichero) y el lenguaje que debe ser escrito.

2. Preguntas

1. (33 puntos) **GradeMidTermExam**
   (directorio: `midtermexam`, fichero: `grademidtermexam.cpp`, lenguaje: C++) Los exámenes de mitad de semestre son muy populares en las universidades del primer mundo. La Universidad ha pensado en eliminar los exámenes parciales y llevar a cabo un examen de mitad de semestre en la semana 8 y otro final en la semana 16.

   Estamos en el semestre 2016-1 y la universidad se apresta hacer sus exámenes de mitad de semestre en la semana 8, la universidad se ha enterado que ha obtenido notas excelentes en el curso de lenguajes de programación del semestre anterior y decide contratarlo para realizar el cálculo de las notas.

   La universidad ha registrado grupos dee las notas en ficheros donde cada linea tiene un nombre corto y dos notas así:

```
$ cat grademidtermexam01.in
Juan 2.3 3.5
Pedro 4.6 3.9
Jose  1.1 4.0
Julian 4.3 4.0
```

   Usted debe imprimir por cada línea: el nombre y el promedio de las notas de cada estudiante y al final de procesar los ficheros un resumen de los estudiantes y el promedio total de todos los estudiantes:

```
$ ./grademidtermexam grademidtermexam01.in
Juan: 2.9
Pedro: 4.25
Jose: 2.55
Julian: 4.15
Total students: 4
Average grade: 3.4625

$ cat grademidtermexam02.in
David 1.5 3.9
Angela 4.6 4.7
Jose  2.0 5.0
```

Al ejecutar con los ficheros de entrada se obtiene:

```
$ ./grademidtermexam grademidtermexam01.in grademidtermexam02.in
Juan: 2.9
Pedro: 4.25
Jose: 2.55
Julian: 4.15
David: 2.7
Angela: 4.65
Jose: 3.5
Total students: 7
Average grade: 3.52857
```

El programa debe procesar todos los ficheros y dar el resumen de todos los estudiantes.

Si no se procesan fichero o estudiantes la salida debe ser:

```
$ ./grademidtermexam
Total students: 0
```

2. (34 puntos) **HelloWorld**
(directorio: `helloworld`, fichero: `HelloWorld.rb`, función: `isHelloWorld` lenguaje: `Ruby`)
Todos sabemos que el profesor de la materia de lenguajes de programación es muy repetitivo con el ejemplo de `"Hola Mundo"`. Los herederos de Dennis Ritchie han decidido ponerle coto[1] al uso sin licencia de la famosa frase, para ello han analizado los ejemplos dados por el docente en twitter y han obtenido listas de los diferentes usos de la frase que han traducido al inglés, pero la traducción no ha sido homogénea puesto que las frases aparecen en mayúsculas y minúsculas, o en algunos casos combinando ambas. Se ha implementado un pequeño proyecto llamado `HelloWorld` donde se ha puesto un fichero `HelloWorld.rb` donde se encuentra un función llamada `isHelloWorld`, esta función recibe una cadena y verifica que dicha cadena sea la cadena `"Hello World"`, si es así la función retorna `true` en caso contrario retorna `false`.

   El proyecto ejecuta se ejecuta así:

```
$ ./processHelloWorld < processHelloWorld01.in
times: 4
$ ./processHelloWorld < processHelloWorld02.in
times: 5
```

   El contenido de `processHelloWorld01.in` es:

```
$ cat processHelloWorld01.in
Hello World
hello world
Hello WORLD
Bye World
Goodbye WORLD
hello WORLD
```

   El contenido de `processHelloWorld02.in` es:

```
$ cat processHelloWorld02.in
hello world
HELLO WORLD
HelLo World
HeLLO WORld
HelLO WorlD
bye world
bye, bye, baby
```

[1] parar.

3. (33 puntos) **Lineas largas**

   (directorio: `longlines` fichero: `shortlonglines.cpp`, lenguaje: C++) Existe una regla establecida para escribir programa en muchos lenguajes es de no pasar de líneas más largas de 80 caracteres. Pero qué pasa si tenemos un título, una cadena de caracteres que ocupa más de 80 caractereso, los lenguajes de programación permite el mecanismo de separación de líneas largas. Muchos lenguajes para separar líneas largas el carácteres (\) al final de una línea para unirla a la siguiente.

   En este problema vamos a implementar el mecanismo inverso, que al leer una secuencia de líneas, el programa una todas la líneas separadas en una sola, por ejemplo:

   ```
   Una\
   Dos\
   Tres
   Cuatro
   ```

   Estas forma dos líneas:

   ```
   UnaDosTres
   Cuatro
   ```

   Si se están uniendo líneas largas y termina la entrada:

   ```
   Una Dos
   ```

   Nos produce:

   ```
   UnaDos
   ```

   Pistas: Mirar la documentación de la clase **string** y buscar los métodos: `append`, `substr`, entre otros.

   La constante \no se puede escribir directamente en el código de C++, tiene que escribirse \\.