UNIVERSIDAD
DE GRANADA

Doble Grado en Informática y Matemáticas

Fundamentos de Programación

Examen práctico 1. Curso 2019/2020

- Cada alumno debe encender un ordenador del aula e insertar su usuario y password junto con el código `examenFPprado`. Esta imagen no tiene acceso a internet (salvo a PRADO y a `decsai.ugr.es`) ni a los puertos USB. Si no pudiera entrar en su cuenta puede utilizar la genérica con el usuario `generica` y password `temporal`.
- Los ficheros han de entregarse a través de la plataforma PRADO. Si no fuera posible acceder a PRADO, deben entregarse en la plataforma docente en `decsai.ugr.es`, o alternativamente en `https://150.214.191.180`.
- Los ficheros a entregar deben llamarse, obligatoriamente, `diamante.cpp`, `orden.cpp`, `numeros.cpp`, `secuencia.cpp`. No se deben comprimir en un archivo `.zip`. Sólo deben entregarse ficheros `.cpp`.
- En la cabecera de cada fichero se deben incluir los siguientes datos, como un comentario del programa:

```
// Nombre: <Nombre y Apellidos del alumno>
// Problema: <diamante u orden o numeros o secuencia>
```

- En la evaluación de los ejercicios se tendrá en cuenta, además de la corrección de la solución propuesta, el estilo de programación, el uso correcto de espacios y tabuladores, así como la claridad del código fuente y de los comentarios. Es decir, se tendrán en cuenta las buenas prácticas de programación explicadas en las transparencias de teoría.
- Para resolver los ejercicios sólo se pueden usarse la sintaxis y herramientas descritas en los temas 1 y 2 de teoría, y los guiones de prácticas realizados hasta la fecha. En particular, NO pueden usarse bibliotecas diferentes a las vistas en clase; ni vectores/arrays, clases, funciones o punteros.
- Los ejercicios se pueden entregar tantas veces como se quiera durante el examen. De hecho, se recomienda que se entreguen varias veces a lo largo del examen ya que, si el ordenador se quedara "colgado", habría que reiniciarlo y se perdería toda la información. Lo ideal es tener el explorador abierto en la página de entrega de PRADO para, cuando se necesite, "soltar" rápidamente el archivo en la zona de subida y actualizar la entrega.
- Este examen se corresponde con un 25 % (1 punto de 4) de la nota de prácticas. Se puede entregar cualquier combinación de los cuatro ejercicios (en particular, los cuatro, ya que PRADO permite subir hasta cuatro archivos). La máxima nota se obtiene si el examen se califica con 10 o más puntos. Los puntos que se obtengan a partir de 10 servirán para compensar posibles fallos en el segundo examen práctico.
- Tiempo de la prueba: 1 hora y 30 minutos (de 15:45 horas a 17:15 horas)

---

1. **Diamante (4 puntos).** Implementar un programa que muestre por pantalla un diamante formado por asteriscos y cuyo interior está formado por guiones. La altura del diamante viene dada por el usuario. Por ejemplo, para altura 7, debería mostrar:

```
   *
  *-*
 *---*
*-----*
 *---*
  *-*
   *
```

   Crear un filtro para que el usuario inserte una altura adecuada.

2. **Orden (3 puntos)**. Se pretende establecer un nuevo orden para comparar valores enteros positivos. Un valor entero a es mayor que otro b si el número de dígitos nueve es mayor en a que en b. Si el número de nueves es igual en los dos, el mayor es el que tiene más ochos. Si hay empate también con este dígito, se considera el siete. Así hasta llegar al cero, si fuese necesario. Si la frecuencia de todos los dígitos es igual en ambos valores, se les considera iguales bajo este orden.

   Por ejemplo, `92341` es menor que `99`, es mayor que `9432000`, y es igual a `12349`.

   Se pide diseñar un programa que, atendiendo al orden descrito anteriormente, dados dos enteros positivos a y b, determine si a es mayor estricto que b, si son iguales, o si b es mayor estricto que a. Suponer que el usuario inserta correctamente números positivos.

3. **Números (3 puntos)**. Escribir un programa que muestre todos los números de tres cifras que se pueden formar con los dígitos 1, 2, 3, 4 y 5 en los que no se repiten dígitos. Por ejemplo, `153` es un número válido, pero `112`, no lo es. También se debe mostrar cuántos son en total.

4. **Secuencia (3 puntos)**. Un satélite de TV digital emite secuencias arbitrariamente largas de caracteres 0's y 1's, hasta un carácter terminador `#`, que representan números en binario (el primer dígito recibido son las unidades, el segundo las decenas, etc.). Para procesarlos correctamente por una SmartTV, cada número se debe transformar a su complemento a dos y después pasarlo a base decimal. Escribir un programa que ayude a realizar el proceso de decodificación de un sólo número, es decir, la entrada será una secuencia de 0's y 1's que termina con un #. Suponer que el satélite emite sólo 0's, 1's y una única # y que se reciben correctamente.

   Nota: El complemento a dos se calcula cambiando 0's por 1's y 1's por 0's, y sumando 1 en binario al resultado.

   Algunos ejemplos:

Entrada:	`0101#`	`0000#`	`11001#`	`101110001#`
Salida:	`6`	`0`	`13`	`227`

   Ampliación 1 (1 punto): Reformar el programa para que se pueda decodificar en una entrada varias secuencias binarias separadas por #, cada una correspondiente a un número, hasta la secuencia de finalización ##.

   Por ejemplo:

Entrada:	`0101#0000#11001#101110001##`
Salida:	`6 0 13 227`

   Ampliación 2 (1 punto): Supongamos que, debido a interferencias, podemos recibir cualquier otro carácter distinto a 0, 1 y #. Reformar el programa para que, en ese caso, corrija lo recibido siguiendo el siguiente algoritmo: calcular el número de ceros y de unos de la codificación ASCII en binario (8 bits) del carácter recibido (suponer caracteres usuales, ASCII entre 32 y 126); si hay más unos, corregimos a 1; si hay más cero, corregimos a 0; si hay el mismo número, corregimos a #. Por ejemplo, si recibimos `w`, se corresponde con el número 119, en binario `01110111`, por lo que se corregiría a `1`.

   Entregar, solamente, la versión más avanzada del Ejercicio 4 que se haya implementado.