Nº Orden	Apellido y nombre	L.U.	Cantidad de hojas

Organización del Computador 2
Recuperatorio del Primer Parcial – 23/11/17

1 (40)	2 (40)	3 (20)	

Normas generales

- Numere las hojas entregadas. Complete en la primera hoja la cantidad total de hojas entregadas.
- Entregue esta hoja junto al examen, la misma **no** se incluye en la cantidad total de hojas entregadas.
- Está permitido tener los manuales y los apuntes con las listas de instrucciones en el examen. Está prohibido compartir manuales o apuntes entre alumnos durante el examen.
- Cada ejercicio debe realizarse en hojas separadas y numeradas. Debe identificarse cada hoja con nombre, apellido y LU.
- La devolución de los exámenes corregidos es personal. Los pedidos de revisión se realizarán por escrito, antes de retirar el examen corregido del aula.
- Los parciales tienen tres notas: I (Insuficiente): 0 a 59 pts, A- (Aprobado condicional): 60 a 64 pts y A (Aprobado): 65 a 100 pts. No se puede aprobar con A- ambos parciales. Los recuperatorios tienen dos notas: I: 0 a 64 pts y A: 65 a 100 pts.

Ej. 1. (40 puntos)

En los partidos de basket suelen contabilizarse estadisticas del juego. En particular, vamos a tener una lista con todos los lanzamientos al aro que se realizaron en el juego, los puntos que produjo el lanzamiento (entre 0 y 3 puntos) y en el orden en el que sucedieron.

La estructura puntos indica los puntos producidos por cada lanzamiento, una breve descripcion indicando el tipo (bandeja, suspendido, vuelco, libre, etc), un booleando indicando si el lanzamiento fue realizado por el equipo local o el visitante y el numero del jugador que realizo dicho intento.

```
typedef struct {
  uint8_t puntos;
  char* descripcion;
  bool local;
  lanzamiento* siguiente;
  uint8_t jugador;
} lanzamiento;
```

Se pide ordenar la lista de lanzamientos al aro de manera que queden primeros los lanzamientos del equipo local y segundo los del equipo visitante. El orden relativo de los lanzamientos del equipo visitante y del local deben mantenerse.

a- (6p) Indicar los desplazamientos dentro de la estructura y su tamanio, notar que no es `__packed__`.

b- (4p) Plantear la aridad de la función a realizar. Justificar el porqué de los parámetros.

c- (12p) Escribir en C la función pedida.

d- (28p) Escribir en ASM la misma funcion pero ademas eliminando los lanzamientos que no produjeron puntos.

Ej. 2. (40 puntos)

```
typedef struct {
  uint2_t lanzamiento;
  uint1_t acierto;
  uint1_t equipo;
  uint12_t jugador;
} lanzamiento __attribute__((packed));
```

Donde se usan dos bits para indicar el tipo de lanzamiento (nulo: 00, libre:01, doble:10 o triple:11), un bit para indicar si el lanzamiento entro en el aro, un bit para indicar si el intento fue realizado por el equipo local o el visitante y, por ultimo, 12 bits para indicar el numero del jugador que realizo el lanzamiento.

El largo de los arreglos que procesaremos tendrán largo múltiplo de 8.

a- (25p) Escribir en ASM una función `void cantidad_de_puntos(lanzamiento* lanzamientos, int tamanio, uint8_t el_equipo)` que devuelva la cantidad de puntos anotados por `el_equipo`.

b- (15p) Escribir en ASM una función `float promedio_de_tres(lanzamiento* lanzamientos, int tamanio)` que calcule el promedio de los lanzamientos de tres del equipo local.

Ej. 3. (20 puntos)

Supongamos que tenemos una funcion recursiva que no utiliza registros de la convencion ni variables locales. Cada tanto, esta funcion recursiva llega a un caso base en el cual, por alguna razon, quiere retroceder no un llamado, sino n llamados.

Se pide implementar la funcion `returnene` que en vez de volver al llamado anterior, vuelva n llamados anteriores. Suponer que esos llamados estan en la pila.

(a) **(5p)** Dibujar el estado de la pila, justo despues de llamar a retornen, con tres llamados recursivos.

(b) **(15p)** Escriba el código ASM de la función `retornen`.
La aridad de la función será: `void returnene(unsigned int n)`