Apellido y Nombres	Legajo	Calificación

PARTE TEÓRICA

1) Considere las siguientes líneas al inicio de un programa:

```
{
 char c = 127;
 int coord [2] = {-52,31};
 char *p;
 p = &c;
```

Sabiendo que:

- el Sistema Operativo asignó al programa la sección de datos que se muestra en la Figura, de modo que las variables se almacenen desde la dirección 0x4000FFD0 en adelante,
- que la CPU utiliza notación Ca2 para representar números negativos,
- y que la CPU trabaja en formato little endian,

se pide escribir en las celdas de memoria los valores hexadecimales que corresponden a las variables indicadas.

0x4C0CFFD0	
0x4C0CFFD1	
0x4C0CFFD2	
0x4C0CFFD3	
0x4C0CFFD4	
0x4C0CFFD5	
0x4C0CFFD6	
0x4C0CFFD7	
0x4C0CFFD8	
0x4C0CFFD9	
0x4C0CFFDA	
0x4C0CFFDB	
0x4C0CFFDC	
0x4C0CFFDD	
0x4C0CFFDE	
0x4C0CFFDF	

2) Considere el siguiente código:

```
int power (int, int);
int main()
{
    power (n ,exp);
    return 0;
}
int power (int num , int e)
{
 int result = num, i;
 for (i =1; i < e ; i ++)
    result*=num;
 return result;
}
```

Escriba en la Figura de la derecha como queda el contenido de la pila cuando inicia el while en la función power. Valen las mismas consideraciones que en el ítem anterior acerca del Sistema Operativo y de la CPU.

0x8FFFC700	
0x8FFFC701	
0x8FFFC702	
0x8FFFC703	
0x8FFFC704	
0x8FFFC705	
0x8FFFC706	
0x8FFFC707	
0x8FFFC708	
0x8FFFC709	
0x8FFFC70A	
0x8FFFC70B	
0x8FFFC70C	
0x8FFFC70D	
0x8FFFC70E	
0x8FFFC70F	
0x8FFFC710	← **Stack Pointer**

3) Señalar cuales de las siguientes acciones son realizadas por el linker (puede haber mas de una):

   a) Empaqueta objetos reubicables en una biblioteca estática.

   b) Genera el objeto ejecutable empaquetando todos los objetos reubicables y resolviendo las referencias entre sus funciones y las funciones existentes en bibliotecas.

   c) Empaqueta objetos reubicables en un Shared Object.

   d) Se encarga de resolver las referencias a funciones de Bibliotecas en tiempo de ejecución del objeto ejecutable.

   e) Genera un objeto reubicable a partir de un programa fuente.

4) Escriba la salida del programa al lado de cada línea de código. Asumir que los datos se almacenan a partir de la dirección de memoria 0x2AD8E600,

```
1  #include <stdio.h>
2  int main ()
3  {
4    unsigned int arr[4]={2500,3500,4500,5500},*p1, *p2;
5    p1 = arr;
6    p2 = &arr[2];
7    printf ("El valor de p1 es: %x\n",p1);
8    printf ("El valor de p es: %x\n",p2);
9    printf ("(p2 - p1) = %d \n",(p2 - p1));
10   printf ("(int) p2 - (int) p1 = %d \n",((int) p2 - (int) p1));
11   printf ("(*p2 - *p1) = %d \n",(*p2 - *p1));
12   *p1 += 6;
13   printf ("*p1 + 6 = %d\n",*p1);
14   printf (" p2 = %x \n",p2);
15   printf ("--p2  d   %x \n",--p2);
16   printf ("p2++  d   %x \n",p2++);
17   printf ("*(p2++)  %d \n",*(p2++));
18   printf ("*p2++  %d \n",*p2++);
19   printf ("p2 = %x \n",p2);
20   return 0;
21 }
```

07 _____ 14 _____

08 _____ 15 _____

09 _____ 16 _____

10 _____ 17 _____

11 _____ 18 _____

13 _____ 19 _____

PARTE PRÁCTICA

Se necesita desarrollar un sistema de moderación de mensajes para aplicar a un sistema de mensajería instantánea interno de una empresa.
El sistema deberá reemplazar con `'X'` cada una de las letras de aquellas palabras consideradas inapropiadas.
La lista de palabras ingresa a la aplicación a desarrollar por línea de comandos a razón de una palabra por argumento.
El mensaje de texto a moderar ingresa por ***stdin***, de modo que se requiere el uso de **scanf** para leer la ***string*** de entrada.
El sistema puede moderar mensajes de a lo sumo 4096 bytes de longitud ('íncluido). Si el input por ***stdin*** supera este tamaño, el programa deberá procesarlo de todos modos dividiéndolo de algún modo en bloques del tamaño soportado o mediante el método que el programador considere conveniente.
La salida del programa debe ser la siguiente: por ***stdout*** debe salir la string moderada, y por ***stderr*** las estadísticas consistentes en las palabras encontradas y la cantidad de veces que se encontraron en el mensaje (el mnsaje de entrada es la totaidad del mismo aunque este supere la capacidad de 4096 del sistema).

CONDICIONES MANDATORIAS:
El proyecto debe contar con:

- **Makefile**. Se provee un frame en el cual debe completar con los parámetros necesarios para que genere el ejecutable
- **Fuentes**. Como mínimo se requiere de los siguientes archivos, cuyos nombres deben ser los siguientes
  - **`./includes/1erParcial.h`**: A crear y completar por el programador de acuerdo a lo que se requiera
  - **`./funcs.c`**: Template provisto con los comentarios en formato doxygen que explican las funciones que se deben desarrollar com parte del requerimiento. Los prototipos de las mismas deben escribirse en el archivo header requerido en el punto anterior.
  - **`./main.c`**: En este archivo se requiere gestionar la correcta invocación del programa en cuanto a la cantidad de argumentos, y preparar la llamada a la función principal del programa. Debe ser lo mas breve posible. El grueso de la aplicación debe estar en el archivo **`funcs.c`**.

RECURSOS DE TESTING:
Se provee un script **`./test.sh`** para probar el programa de manera ágil. Se recomienda darle permisos de ejecución al descargarlo en su máquina.
Se proveen dos archivos para emplear en el test del sistema a desarrollar:
`DiscursoDelMetodoDescartes.txt`, que contiene el texto a moderar, y ***`blacklist`*** con la lista de palabras no aceptadas y que deben reemplazarse por 'X'cada vez que se las detecte.