1ER PARCIAL

1-

```
/* EJERCICIO:

Escribir una función que transponga una matriz de NxN. El prototipo debe ser:
void transponer (double mat[N][N]), siendo N una constante ya definida. El
resultado debe devolverse modificando la matriz original.

La traspuesta de una matriz se obtiene reflejando los elementos a lo largo de
su diagonal.

*/
```

RTA

```
/

// Función que transpone una matriz de double[N][N]
void transponer (double mat[N][N])
{
    int i,j;
    double aux;

    1   2   3
    4   5   6
    7   8   9

    for (i=0 ; i<N-1 ; ++i)
    {
        for (j=i+1 ; j<N ; ++j)
        {
            aux = mat[i][j];
            mat[i][j] = mat[j][i];
            mat[j][i] = aux;
        }
    }
}
```

2- Indicar y justificar que imprime el siguiente programa

```c
{
    char *p = text + 20;

    myprint(p+5);
    myprint(&p[16]);
    myprint(p+30);
    myprint(&text[13]);
    myprint(text+46);

    return 0;
}

void myprint(char *p)
{
    while(*p != ' ' && *p)
        putchar(*p++);
    putchar('\n');
}
```

Rta

```c
/* SOLUCIÓN:

myprint imprime un string hasta encontrar un espacio ' ' o un terminador \0',
y agrega un '\n' al final.
p apunta a text+20

it       -> text+25, que es "it rain..", imprime "it\n"
sn't     -> text+36, que es "sn't wo..", imprime "sn't\n"
<basura> -> text+50, está fuera del string, imprime basura hasta encontrar un
' ' o '\0' y agrega un '\n' al final
car      -> text+13, que es "car to..", imprime "car\n"
         -> text+46, es justo el terminador (fin del string), solo imprime el
         "\n"

*/
```

3-

```c
/* EJERCICIO:

Utilizando directivas de precomplilación, crear una constante ARQ_SIZE que sea
un texto que explique el tamaño de la arquitectura.
Por ejemplo, en una arquitectura de 32 bits, el texto sería <Arquitectura de
32 bits>

*/

#if sizeof(int) == 1
#define ARQ_SIZE   "Arquitectura de 8 bits"
#elif sizeof(int) == 2
#define ARQ_SIZE   "Arquitectura de 16 bits"
#elif sizeof(int) == 4
#define ARQ_SIZE   "Arquitectura de 32 bits"
#elif sizeof(int) == 8
#define ARQ_SIZE   "Arquitectura de 64 bits"
#endif
```

4-

```
/*****************************************************************************/

/* EJERCICIO:

Se tiene un archivo main.c. Escribir la secuencia en línea de comando (CLI)
para:
a) Compilarlo utilizando gcc, de manera que genere todos los errores,
definiendo la constante DEBUG, agregando la información para poder debuggearlo
con GDB y que el archivo ejecutable se llame lectura.
b) Ejecutarlo, asignado como stdin el archivo diccionario.txt y como stdout la
consola/terminal.

*/


/* SOLUCIÓN:

~$ gcc main.c -Wall -o lectura -g -D DEBUG

~$ ./lectura < diccionario.txt
```

5- Reescribir el siguiente programa usando while, sin continue ni break y que siga haciendo lo mismo:

```
/* SOLUCIÓN:
*/


int fun (int x)
{
    int i=1, y=0;

    while (i*i < x*x) // o ((i*i >= x*x) == 0)
    {
        if (i%2)
        {
            y += i*i;
        }
        ++i;
    }

    return y;
}
```