Solucion sección teórica

1. C, Se puede producir un desbordamiento de pila
2. D,
3. D, no compila ya que el operador % no se puede usar sobre una variable tipo float

Solución sección práctica

1.
```
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>

unsigned short *companding(unsigned char *p, unsigned int cant)
{
unsigned int i;
unsigned short *salida;
unsigned short entrada, offset;
const unsigned short tabla[] = {0x0080, 0x0100, 0x0200, 0x0400, 0x0800,
0x1000, 0x2000, 0x4000};

    if ((p == NULL) || (cant == 0)) {
        return (NULL);
    }

    salida = malloc(cant * sizeof(*salida));
    if (salida == NULL) {
        return (NULL);
    }
    for (i = 0; i < cant; i++) {
        entrada = (unsigned short)*(p + i);
        offset = (entrada >> 4) & 0x07;
        *(salida + i) = ((entrada & 0x80) << 8) | tabla[offset] |
((entrada & 0x000F) << (3 + offset));
    }

    return salida;
}

int main (void)
{
unsigned char datos[] = {0b10000000, 0b00001001, 0b00011001,
0b01111001};
unsigned short *res;
unsigned int i;

    res = companding(datos, sizeof(datos) / sizeof(datos[0]));

    for (i = 0; i < sizeof(datos) / sizeof(datos[0]); i++) {
        printf("Salida[%u] = %016B\n", i, *(res + i));
    }

    free(res);
    return 0;
}
```

2.

```
#include <fcntl.h>
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
#include <sys/stat.h>

struct rle_S {
    char caracter;
    int cantidad;
} ;

struct rle_S* rle(char *archivo, int *cant)
{
int fd;
struct stat st;
char *p;
ssize_t cantRead;
struct rle_S *salida = NULL, *aux;
ssize_t i;
int index, count;

    if ((archivo == NULL) || (cant == NULL)) {
        return (NULL);
    }

    *cant = 0;
    fd = open(archivo, O_RDONLY);
    if (fd < 0) {
        return (NULL);
    }

    if (fstat(fd, &st) < 0) {
        close(fd);
        return (NULL);
    }

    if (st.st_size == 0) {
        close(fd);
        return (NULL);
    }

    p = (char*)malloc(st.st_size);
    if (p == NULL) {
        close(fd);
        return (NULL);
    }

    cantRead = read(fd, p, st.st_size);
    if (cantRead != st.st_size) {
        free(p);
        close(fd);
        return (NULL);
    }

    close(fd);

    //-- Proceso los datos --
    index = 0;
    count = 1;
    for (i = 1; i < cantRead; i++) {
        if (*(p + i) == *(p + i - 1)) {
            count++;
        } else {
            aux = realloc(salida, (index + 1) * sizeof(*aux));
            if (aux == NULL) {
                free(p);
                    if (salida!= NULL) {
                    free (salida);
                }
```

```
                return (NULL);
            }

            salida = aux;
            (salida + index)->caracter = *(p + i - 1);
            (salida + index)->cantidad = count;
            index++;
            *cant = index;
            count = 1;
        }
    }

    aux = realloc(salida, (index + 1) * sizeof(*aux));
    if (aux == NULL) {
        free(p);
        if (salida!= NULL) {
            free (salida);
        }
        return (NULL);
    }
    salida = aux;
    (salida + index)->caracter = *(p + i - 1);
    (salida + index)->cantidad = count;
    index++;
    *cant = index;

    free(p);
    return (salida);
}

int main() {
int cantidad;
struct rle_S *resultado;

    resultado = rle("texto.txt", &cantidad);
    if (resultado != NULL) {
        for (int i = 0; i < cantidad; i++) {
            printf("'%c' -> %d veces\n", resultado[i].caracter, resultado[i].cantidad);
        }
        free(resultado);
    }

    return 0;
}
```