# Parcial 1. TDL.

**1**

```
Linea = posible.
L = Nuevo texto.
Linea = Nuevo te.
```

Como le asigna linea + 11 = NADA entonces se corta en destino + 8 ?

**2**

i = 10 → 5

10 9 8 7 6

**3**

```
#define nPares(a,b) (!((a)%2)+!((b)%2))
```

**4**

a) F  b) Agregándole el Printf, V  c) V

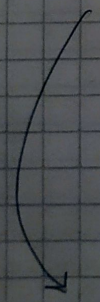
d) F  e) F  f) F  g) V  h) V

i) F  j) V

A = XOR

```
0010
0101
----
0111
```

**5**



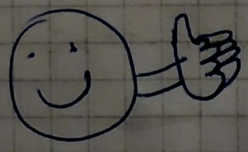
```c
#include <stdio.h>
#include <stdlib.h>

typedef struct destino {
    int codProv;
    int codLoc;
    char nomLoc[30];
    int hab;
} Destino;

int main() {
    FILE *arch = fopen("Habitantes.txt", "r");
    FILE *barch = fopen("Habitantes.dat", "wb");
    if ((arch == NULL) || (barch == NULL))
        return 1;
    Destino d;
    fscanf(arch, "%d %d %s %d", &d.codProv, &d.codLoc, d.nomLoc, &d.hab);
    while (!feof(arch)) {
        fwrite(&d, sizeof(Destino), 1, barch);
        fscanf(arch, "%d %d %s %d", &d.codProv, &d.codLoc, d.nomLoc, &d.hab);
    }
    fclose(arch);
    fseek(barch, 0, SEEK_SET);
    Destino dMax;
    fread(&d, sizeof(Destino), 1, barch);
    dMax = d;
    while (!feof(barch)) {
        if (d.hab > dMax.hab)
            dMax = d;
        fread(&d, sizeof(Destino), 1, barch);
    }
    printf("%d %s", dMax.codLoc, dMax.nomLoc);
    fclose(barch);
    return 0;
}
```

ES MEJOR HACER UNA ESTRUCTURA DE DATOS Y NO ACCEDER SIEMPRE.

¡Funciona!



⑥

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main(int argc, char *argv[]) {
    if (argc == 1)
        Printf("Error");
    else {
        for(int i=1; i<argc; i++){
            Printf("%s ", argv[i]);
        }
    return 0;
}
```

Se Pudo

```c
#include <stdio.h>
#include <stdlib.h>
```

```c
#define N 3
void imprimir(int *m, int n);
```

7)

```c
#include <stdio.h>
#include <stdlib.h>
#define N 3

int ** reservar(int n);
void inicializar(int** m, int n);
void imprimir(int ** m, int n);
void liberar(int **m, int n);
int main() {
    srand(time(null));
    int **m = reservar(N);

    inicializar(m, N);

    imprimir(m, N);

    liberar(m, N);
                ↳ liberar(m, N);
    return 0;
}
```

```c
int** reservar(int n) {
    int ** m = malloc(n * sizeof(int*));

    for(int i=0; i<n; i++) {
        m[i] = malloc((i+1) * sizeof(int));
    }
    return m;
}

void inicializar(int **m, int n) {
    for(int i=0; i<n; i++) {
        for(int j=0; j<(i+1); j++) {
            m[i][j] = rand() % 21;
        }
    }
}
```

```c
void imprimir(int ** m, int n) {
    for(int i=0; i<n; i++) {
        for(int j=0; j<i+1; i++){
            Printf("%d", m[i][j]);
        }
        Printf("\n");
    }
}
```

```c
void liberar(** m, int n) {
    for(int i=0; i<n; i++) {
        free(*m[i]);        free(*m[i]);
    }
    free(*m);
}
```

verificar.