1) **Mit ír ki?** 6 pont

Mindhárom kódrészletnél az a feladat, hogy mondjuk meg mit írnak ki. Azonban, ha a kódban hiba van, ami miatt nem lefordítható, vagy nem tudja befejezni a futását, vagy "undefined behaviour" következik a hibából, akkor karikázzuk be a kód hibás részét, és írjunk egy pár szavas magyarázatot a hibáról.

a)
```
float tomb[] = {1.1, 2.2, 3.3, 4.4};
int i;
for(i = 1; i <= 4; ++i) {
    printf("%.2f", tomb[i]*2);
}
```

b)
```
int a = 10;
int b = 20;
int *m = &a;
// Az ilyen printf 3 szamot ir ki szokozzel elvalasztva:
printf("%d %d %d\n", a, b, *m);
*m += 5;
m = &b;
printf("%d %d %d\n", a, b, *m);
b += 5;
printf("%d %d %d\n", a, b, *m);
```

c)
```
void kiir(int a) {
    switch(a) {
        case 1:
            printf("egy\n");
            break;
        case 5:
            printf("ot\n");
            break;
        case 4:
            printf("negy\n");
            break;
        default:
            printf("valami mas\n");
            break;
    }
}

int main() {
    int i;
    for(i = 1; i < 10; i *= 2) {
        kiir(i);
    }
}
```

2) **Kód írás** 9 pont

a) Alap szintaxis (3 pont)

Írjuk le egy C függvény teljes definícióját! A függvény neve legyen `pozitivak`, visszatérési típusa egész szám, és két paramétere legyen. Az első paraméter egy egész számokból álló tömb, a második paraméter ennek a tömbnek a hossza. A függvény egy egész számmal térjen vissza, a tömbben található pozitív számok számával.

b) Kiegészítés (6 pont)

Egészítsük ki a kódot, hogy megoldja a feladatot. Arisztotelész és Baltazár minden nap süt egy-egy nagy tortát. A program N napon kereszül kövesse az eseményeket, és minden napra mondja meg a nagyobb torta méretét. A program parancssorról olvassa be először N-et, majd Arisztotelész tortáinak a méretét mind az N napra, majd Baltazár tortáinak méretét mind az N napra. A program N számot írjon ki 2 tizedesjegy pontossággal, mind az N naphoz a nagyobb torta méretét. A méretek valós számok (nem feltétlen egészek). N mindig legfeljebb 100.

A már megírt kód beolvassa N-et, és Arisztotelész tortáinak a méretét az a tömbbe. Emlékeztető: a valós számok kiírásánál a következő módon lehet megadni a formátumot: ha azt írjuk hogy `"%X.Yf"`, az azt jelenti, hogy a teljes kiírt szám legalább X karaktert foglaljon, és Y tizedesjegy pontossággal írja ki a számot. Mindkettő elhagyható külön külön, de az Y-nak mindenképpen egy pont után kell lennie.

Példa felhasználás a következő oldal elején!

```
int main() {
    float a[100];

    int N;
    int i;
    scanf("%d", &N);
    for(i = 0; i < N; ++i) {
        scanf("%f", &a[i]);
    }

    return 0;
}
```

Ha jól működik a program, erre a bemenetre:

```
5
1.2345 4.56 3 0.003 007
2.3456 4.45 9 0.002 4.2
```

Azt írja ki hogy:

```
2.35 4.56 9.00 0.00 7.00
```

3) **Elmélet** 10 pont

a) Mik az előnyei a **lefordított** programozási nyelveknek az **interpretáltakhoz** képest? Soroljunk fel kettőt! (2 pont)

-
-

b) Írjatok ide **egy sor** C kódot, ami egy karakterlánc változót definiál. A változó neve legyen **udvozles**, és a tartalma valamilyen köszönés legyen. (1 pont)

c) Mire használhatóak a C-ben az unáris (egy paraméterű) "*" (csillag) és "&" ("és" jel) operátorok? (2 pont)

d) Amikor egy mutatót definiálunk, kétféleképp is konstanssá tehetjük, attól függően hogy hova írjuk hogy `const`. Mi a különbség a kettő között? (1 pont)

```
<const?> int * <const?> mutato = &a;
```

e) Meddig tart egy veremen létrejött változó élettartama? (1 pont)

f) Hogy hívják az automatikus memóriakezelési módszert, amit a python használ? (1 pont)

g) Mire használható a speciális beépített `free()` függvény? Milyen paramétert vár? (2 pont)