1. Zárthelyi dolgozat

1. A következő kód feladata a kapott egész szám összes prím osztójának kiírása. Van benne 2 logikai hiba. Mik ezek? (Nem kell kijavítani a kódot, elég 1-1 mondatban megfogalmazni a hibákat.) (*10 pont*)

```
#include<iostream>
using namespace std;

void prim(int n) {
  for(int i = 2; i < n; i++) {
    if(n % i == 0) {
      cout << false;
    }
  }
  cout << true;
}
void osztok(int n) {
  for(int i = 2; i <= n; i++) {
    if(prim(n)) {
      cout << i << " ";
    }
  }
}
int main(void) {
  int n;
  cin >> n;
  osztok(n);
  return 0;
}
```

2. Írjunk *szelet* névvel függvényt, mely a kapott C string-et az első karaktertől a kapott indexű karakterig lemásolja és ezt az új C stringet visszaadja.(*10 pont*)

   - *str* a string amiből másolunk, char tömb
   - *utolso* az utolsó index amit még másoljunk, sima int

   Például a `szelet("kiskutya", 2)` adja vissza a `"kis"` string-et.

3. Írjunk függvényt *max* névvel mely visszaadja a kapott *float* tömb legnagyobb abszolút értékű elemét és az indexét is. A bemenetei:(*10 pont*)

   - *tomb*, a float tömb
   - *hossz* a tömb hossza
   - Megoldástól függően még lehetnek paraméterek.

4. Írjunk egy Vector osztályt mely képes tetszőleges dimenziós vektort reprezentálni. (*10 pont*)

   - Legyen konstruktora, ami a kapott float tömbből konstruálja a vektort.
   - Legyen konstruktora, ami csak egy egész számot kap és ekkora csupa 0 vektort konstruál.
   - Legyen *scalar* metódusa, ami skalárral szorozza a vektort.
   - Ha szükséges, akkor legyen destruktora.

   Implementáljuk az egyik konstruktort (tetszőleges), a scalar-t és a destruktort.

5. Adott a szokásos láncolt listánk: (*10 pont*)

```
struct list_e {
  int num;
  struct list_e *next;
};
```

Írjunk függvényt *utolso* névvel mely az utolsó elem értékét módosítja. A bemenetei:

- *start*, a láncolt lista első elemének pointere, struct list_e*.
- *ertek*, az érték amire módosítja az utolsó elem tárolt értékét, sima int.