**Facultatea de Matematică și Informatică**

**Universitatea din București**

**Laborator**

**Programare orientată pe obiecte**

**Proiectul 1 - Tema 3**

**Ștefan Râpeanu-Andreescu**

|  |  |  |
| --- | --- | --- |
| **Tema 3** | | **Număr întreg mare** |
| **1.4 p** | **Toate cerințele generale, valabile pentru toate temele** | |
| **0.8 p** | **Respectarea unor standarde de codare** | |
| 0.1 p | x | Scrierea declarației clasei într-un fișier header cu extensia hpp și scrierea definiției metodelor clasei și a funcțiilor prieten într-un fișier sursă cu extensia cpp. |
| 0.1 p | x | Denumirea fișierului header și a fișierului sursă cu numele clasei pe care o conțin. |
| 0.1 p | x | Utilizarea consistentă a unei convenții de denumire a atributelor clasei care să ajute la ușoara identificare a acestora în cod. Convenția utilizată poate fi oricare dintre următoarele:  - "m\_" urmat de denumirea atributului, care să înceapă cu literă mică (Ex: m\_attribute);  - "\_" urmat de denumirea atributului, care să înceapă cu literă mică (Ex: \_attribute);  - "m" urmat de denumirea atributului, care să înceapă cu literă mare (Ex: mAttribute). |
| 0.1 p | x | Utilizarea consistentă a unei convenții de denumire a atributelor, respectiv a funcțiilor. Convenția utilizată poate fi oricare dintre următoarele:  - Pascal Case: (Ex: LongFunctionName, LongAttributeName);  - Camel Case: (Ex: longFunctionName, longAttributeName);  - Snake Case: (Ex: long\_function\_name, long\_attribute\_name). |
| 0.1 p | x | Indentarea corectă și consistentă a codului. |
| 0.1 p |  | Utilizarea unor nume sugestive pentru variabile și funcții în locul unora generice. |
| 0.1 p | x | Respectarea ordinii declarațiilor în clasă. Mai întâi membrii cu access public, apoi cei cu acces protected și apoi cei cu acces private. Pentru fiecare dintre aceste trei secțiuni ordinea trebuie să fie:  - Definiții de tipuri și enumerări;  - Atribute constante ale clasei;  - Constructori;  - Destructori;  - Metode;  - Atribute neconstante ale clasei. |
| 0.1 p | x | Utilizarea unei singure limbi pentru scrierea codului. Este preferabilă limba engleză, deoarece cuvintele cheie ale limbajului sunt în limba engleză și se păstrează o consistență a codului în acest fel. |
| **0.6 p** | **Versionarea codului** | |
| 0.5 p | x | Versionarea codului pe GitHub. |
| 0.1 p | x | Utilizarea unor mesaje sugestive pentru modificările făcute într-un commit. |
| **9.55 p** | **Toate cerințele individuale ale temei** | |
| **0.6 p** | **Atributele clasei și metodele care permit accesul la acestea** | |
| 0.1 p | x | Implementarea unui atribut privat de tip std::vector<char> pentru stocarea cifrelor numărului. |
| 0.1 p | x | Implementarea unui atribut privat pentru stocarea semnului numărului. |
| 0.1 p | x | Implementarea unui atribut privat pentru stocarea bazei numărului. |
| 0.1 p | x | Implementarea unei metode publice pentru furnizarea bazei numărului. |
| 0.1 p | x | Implementarea unei metode publice pentru furnizarea semnului numărului. |
| **1 p** | **Constructori și destructori** | |
| 0.5 p | x | Implementarea unui constructor parametrizat cu valori implicite pentru parametri. Numărul întreg mare implicit va fi considerat ca fiind 0 în baza 10, iar constructorul trebuie să îndeplinească această cerință. |
| 0.5 p | x | Implementarea unui constructor parametrizat care să primească ca parametru un număr întreg sub formă de std::string și să construiască obiectul pe baza acestuia. Baza va fi considerată implicit ca fiind 10. |
| **3.2 p** | **Supraîncărcarea operatorilor compuși de atribuire** | |
| 0.8 p | x | Supraîncărcarea operatorului binar += ca metodă publică a clasei, care realizează operația de adunare a două numere întregi mari, urmată de atribuirea rezultatului operandului stâng și returnarea unei referințe la operandul stâng. Operatorul trebuie scris în două variante pentru a putea funcționa cu operanzi de tipul (BigInteger, BigInteger) și (BigInteger, int). |
| 0.8 p | x | Supraîncărcarea operatorului binar -= ca metodă publică a clasei, care realizează operația de scădere a două numere întregi mari, urmată de atribuirea rezultatului operandului stâng și returnarea unei referințe la operandul stâng. Operatorul trebuie scris în două variante pentru a putea funcționa cu operanzi de tipul (BigInteger, BigInteger) și (BigInteger, int). |
| 0.8 p | x | Supraîncărcarea operatorului binar \*= ca metodă publică a clasei, care realizează operația de înmulțire a două numere întregi mari, urmată de atribuirea rezultatului operandului stâng și returnarea unei referințe la operandul stâng. Operatorul trebuie scris în două variante pentru a putea funcționa cu operanzi de tipul (BigInteger, BigInteger) și (BigInteger, int). |
| 0.8 p | x | Supraîncărcarea operatorului binar /= ca metodă publică a clasei, care realizează operația de împărțire a două numere întregi mari, urmată de atribuirea rezultatului operandului stâng și returnarea unei referințe la operandul stâng. Operatorul trebuie scris în două variante pentru a putea funcționa cu operanzi de tipul (BigInteger, BigInteger) și (BigInteger, int). |
| **0.2 p** | **Supraîncărcarea operatorilor aritmetici unari** | |
| 0.1 p | x | Supraîncărcarea operatorului unar + ca funcție prieten a clasei, care returnează numărul însuși. |
| 0.1 p | x | Supraîncărcarea operatorului unar - ca funcție prieten a clasei, care returnează opusul numărului. |
| **1.5 p** | **Supraîncărcarea operatorilor aritmetici binari** | |
| 0.3 p | x | Supraîncărcarea operatorului binar + ca funcție prieten a clasei, care realizează operația de adunare a două numere întregi mari și returnează rezultatul. Operatorul trebuie scris în trei variante pentru a putea funcționa cu operanzi de tipul (BigInteger, BigInteger), (int, BigInteger) și (BigInteger, int). |
| 0.3 p | x | Supraîncărcarea operatorului binar - ca funcție prieten a clasei, care realizează operația de scădere a două numere întregi mari și returnează rezultatul. Operatorul trebuie scris în trei variante pentru a putea funcționa cu operanzi de tipul (BigInteger, BigInteger), (int, BigInteger) și (BigInteger, int). |
| 0.3 p | x | Supraîncărcarea operatorului binar \* ca funcție prieten a clasei, care realizează operația de înmulțire a două numere întregi mari și returnează rezultatul. Operatorul trebuie scris în trei variante pentru a putea funcționa cu operanzi de tipul (BigInteger, BigInteger), (int, BigInteger) și (BigInteger, int). |
| 0.3 p | x | Supraîncărcarea operatorului binar / ca funcție prieten a clasei, care realizează operația de împărțire a două numere întregi mari și returnează rezultatul. Operatorul trebuie scris în trei variante pentru a putea funcționa cu operanzi de tipul (BigInteger, BigInteger), (int, BigInteger) și (BigInteger, int). |
| 0.3 p | x | Supraîncărcarea operatorului binar ^ care realizează operația de ridicare a unui număr întreg mare la o putere număr natural și returnează rezultatul. |
| **1.8 p** | **Supraîncărcarea operatorilor relaționali** | |
| 0.3 p | x | Supraîncărcarea operatorului == ca funcție prieten a clasei, care să verifice egalitatea a două numere întregi mari. Operatorul trebuie scris în trei variante pentru a putea funcționa cu operanzi de tipul (BigInteger, BigInteger), (int, BigInteger) și (BigInteger, int). |
| 0.3 p | x | Supraîncărcarea operatorului != ca funcție prieten a clasei, care să verifice neegalitatea a două numere întregi mari. Operatorul trebuie scris în trei variante pentru a putea funcționa cu operanzi de tipul (BigInteger, BigInteger), (int, BigInteger) și (BigInteger, int). |
| 0.3 p | x | Supraîncărcarea operatorului < ca funcție prieten a clasei, care să verifice dacă un număr întreg mare este mai mic decât un alt număr întreg mare. Operatorul trebuie scris în trei variante, pentru a putea funcționa cu operanzi de tipul (BigInteger, BigInteger), (int, BigInteger) și (BigInteger, int). |
| 0.3 p | x | Supraîncărcarea operatorului <= ca funcție prieten a clasei, care să verifice dacă un număr întreg mare este mai mic sau egal cu un alt număr întreg mare. Operatorul trebuie scris în trei variante, pentru a putea funcționa cu operanzi de tipul (BigInteger, BigInteger), (int, BigInteger) și (BigInteger, int). |
| 0.3 p | x | Supraîncărcarea operatorului > ca funcție prieten a clasei, care să verifice dacă un număr întreg mare este mai mare decât un alt număr întreg mare. Operatorul trebuie scris în trei variante, pentru a putea funcționa cu operanzi de tipul (BigInteger, BigInteger), (int, BigInteger) și (BigInteger, int). |
| 0.3 p | x | Supraîncărcarea operatorului >= ca funcție prieten a clasei, care să verifice dacă un număr întreg mare este mai mare sau egal cu un alt număr întreg mare. Operatorul trebuie scris în trei variante, pentru a putea funcționa cu operanzi de tipul (BigInteger, BigInteger), (int, BigInteger) și (BigInteger, int). |
| **1.25 p** | **Supraîncărcarea operatorilor de cititre și de scriere** | |
| 0. 5 p | x | Supraîncărcarea operatorului binar >> ca funcție prieten a clasei, care realizează citirea unui număr întreg mare de la tastatură sau dintr-un fișier. Formele de reprezentare a unui număr întreg mare pentru citire sunt "a", "-a". |
| 0.75 p | x | Supraîncărcarea operatorului binar << ca funcție prieten a clasei, care realizează scrierea unui număr întreg mare în consolă sau într-un fișier. Formele de reprezentare a unui număr întreg mare pentru scriere sunt "a", "-a". |