Laboratorium 4

Zadanie 1.

Dana jest klasa Data:

```
class Data {
  int dzien;
  int mies;
  int rok;
public:
  Data(); // konstruktor
  void UstawDate(int d, int m, int r); // ustawia date
};
```

a) Uzupełnij definicje konstruktora i metody `UstawDate()`.
b) Uzupełnij klasę o funkcje dostępowe.

Napisz program, który pozwoli wprowadzić dwie daty i wyświetlić je.

Zadanie 2.

Uzupełnij program o przeciążony operator porównywania == .
Napisz trzy wersje programu:
a) Wykorzystaj funkcje dostępowe.
b) Wykorzystaj zaprzyjaźnianie.
c) Wykorzystaj funkcję sk.ładową.

Zadanie 3.

Uzupełnij program o przeciążone operatory porównywania: >, >=, < , <= , == , != .

Zadanie 4.

Uzupełnij program o przeciążony operator wyprowadzania daty.

Zadanie 5.

Zbuduj klasę `StosLiczb`, która będzie obsługiwała stos liczb całkowitych. Liczby mają być zapamiętywane w dynamicznie przydzielanym obszarze określanym w momencie definiowania stosu. Klasa ma zawierać następujące operatory:

<< operator wkładania na stos, na przykład: `s << n` wkłada liczbę `n` na stos. Jeśli stos jest pełny nic nie robi.

>> operator zdejmowania ze stosu, na przykład `s >>` n zdejmuje ze stosu s liczbę i umieszczają w `n`. Jeśli stos jest pusty, `n` otrzymuje wartość 0.

Operatory `<<` i `>>` powinny dawać się użyć w instrukcjach o postaci:

```
s << n1 << n2 << n3;  // włożenie kilku elementów na stos
c>> n1 >> n2 >> n3;   // zdjęcie kilku elementów ze stosu
```

Prototyp klasy:

```
class StosLiczb
{
  int max;      // maksymalna liczba wartości na stosie
  int lb_elem;  // bieżąca liczba wartości na stosie
  int *stos;    // wskaźnik do przydzielonego obszaru
public:
  StosLiczb(int n=0);       // konstruktor
  ~StosLiczb();             // destruktor
  StosLiczb(StosLiczb & s); // konstruktor kopiujący
  void operator=(StosLiczb &s);  // przypisanie – wyświetlenie komunikatu,
                                 //że operacja nie jest dostępna i
                                 // zakończy wykonywanie programu
  StosLiczb & operator<<(int n);   // wstawianie na stos
  StosLiczb & operator>>(int & n); // zdejmowanie ze stosu
.....
  inne użyteczne funkcje
}
```

Napisz funkcję testującą opracowaną klasę.

Zadanie 6.

Uzupełnij opracowaną klasę o inne użyteczne funkcje, na przykład sprawdzenie czy stos jest pusty lub pełny, zwrócenie informacji o liczbie elementów znajdujących się na stosie, wyświetlenie zawartości stosu (przeciąż operator >>, na przykład `cout << s` wyprowadza zawartość całego stosu), odwrócenie stosu, skopiowanie stosu do innego stosu.

Zadanie 7.

Zmodyfikuj zadanie ze stosem tak, aby program działał na wyrazach. Wyrazy powinny być pobierane z pliku.

Zadanie 8.

Dany jest plik z ocenami studentów. Każdy student opisany jest za pomocą nazwiska i ocen z 5 przedmiotów. Dane studenta są wczytywane z pliku. Napisz program, który obliczy dla każdego studenta średnią ocenę i wyświetli wykaz tych ocen posortowanych według nazwisk. Opracuj potrzebną do tego zadania klasę `StudentInfo`.