Zadanie 1 (na 3.0). Napisz szablony funkcji:

- *clamp* przyjmującą trzy argumenty, wartość minimalną *a*, maksymalną *c* oraz liczbę *b*, zwracającą:

$$clamp(a,b,c)=\begin{cases} a, b<a \\ c, b>c \\ b, a\leq b\leq c \end{cases}$$

- *clamp* która zamiast na jednej liczbie, operuje na tablicy w stylu C (modyfikuje każdy jej element),
- *avg* liczącą średnią arytmetyczną elementów przesłanej tablicy,
- *bubblesort* sortującą podaną tablicę algorytmem bąbelkowym.

Funkcje powinny przyjmować argumenty dowolnego typu, np. sortowanie powinno działać dla tablicy elementów typu *int* oraz typu *Fraction*.

Zadanie 2 (na 4.0). Używając szablonów, zmodyfikuj:

- Klasę stosu z poprzednich zajęć tak, aby mogła przechowywać obiekty dowolnego typu,
- Klasę ułamka zwykłego tak, aby licznik i mianownik mogły być dowolnego typu (np. *char*, *short*, *int*, *long long* itp.).

Stwórz stos 10 losowych elementów typu *int*, wypisz jego zawartość.
Stwórz stos 10 losowych elementów typu *Fraction*<*int*>, wypisz jego zawartość.
Stwórz stos 10 losowych elementów typu *Fraction*<*char*>, wypisz jego zawartość.

Zadanie 3 (na 5.0). Napisz szablon klasy *TypeName* posiadający publiczną statyczną metodę *print* wypisującą „nieznany typ”. Napisz częściową specjalizację klasy *TypeName* dla:

- wskaźnika na typ szablonowy, w którym metoda *print* wypisuje „wskaźnik na ” oraz wywołuje metodę *print* z *TypeName*<*T*>,
- referencji na typ szablonowy, w którym metoda *print* wypisuje „referencja na ” oraz wywołuje *print* z *TypeName*<*T*>,
- stałego typu szablonowego, w którym metoda *print* wypisuje „staly ”oraz wywołuje *print* z *TypeName*<*T*>,
- tablicy obiektów typu szablonowego, w którym metoda *print* wypisuje „tablica ” oraz wywołuje *print* z *TypeName*<*T*>,
- funkcji zwracającej typ *T* i przyjmującej argument *U*, w którym metoda *print* wypisuje „funkcja przyjmujaca ”, następnie wywołuje *print* z *TypeName*<*U*>, wypisuje „zwracająca ” i wywołuje *print* z *TypeName*<*T*>.

Napisz specjalizacje klasy *TypeName* w których metoda *print* wypisuje nazwy dla typów:

- *char*,
- *short*,
- *int*,
- *float*,
- *double*,
- *void*,
- *long long*.

Zaprezentuj działanie:

```
int main(int, char**)
          {
          TypeName<char>::print(); printf("\n");
          TypeName<char *>::print(); printf("\n");
          TypeName<char &>::print(); printf("\n");
          TypeName<const int *>::print(); printf("\n");
          TypeName<const int &>::print(); printf("\n");
          TypeName<void * const>::print(); printf("\n");
          TypeName<const short * const>::print(); printf("\n");
          TypeName<const short * const &>::print(); printf("\n");
          TypeName<char[]>::print(); printf("\n");
          TypeName<int (*)(float)>::print(); printf("\n");
          printf("\n");

          TypeName<const long long * const * const *>::print(); printf("\n");
          TypeName<float******************************>::print(); printf("\n");
          TypeName<int (*)(const char*)>::print(); printf("\n");
          TypeName<const double * const (*)(char* const *&)>::print(); printf("\n");
          TypeName<char (*)(const char * const)>::print(); printf("\n");
          TypeName<char (*[])(const char * const)>::print(); printf("\n");
          TypeName<int (*)(int[])>::print(); printf("\n");


          return 0;
          }
```