Завдання №1

1.1 Напишіть клас з ім’ям Point. У класі Point повинні бути дві змінні-члени типу double: `m_a` і `m_b` зі значеннями за замовчуванням `0.0`. Напишіть конструктор для цього класу і функцію виводу print().

1.2 Тепер додамо метод distanceTo(), який прийматиме другий об’єкт класу Point в якості параметра і обчислюватиме відстань між двома об’єктами. Враховуючи дві точки (a1, b1) і (a2, b2), відстань між ними можна обчислити наступним чином: `sqrt((a1 - a2) * (a1 - a2) + (b1 - b2) * (b1 - b2))`. Функція sqrt() знаходиться в заголовку cmath.

1.3 Змініть функцію distanceTo() з методу класу в дружню функцію, яка прийматиме два об’єкти класу Point в якості параметрів. Перейменуйте цю функцію на distanceFrom().

Завдання №2

Генератор випадкових монстрів.

a) Спочатку створіть перерахування MonsterType з наступними типами монстрів: `Dragon`, `Goblin`, `Ogre`, `Orc`, `Skeleton`, `Troll`, `Vampire` і `Zombie` + додайте `MAX_MONSTER_TYPES`, щоб мати можливість підрахувати загальну кількість всіх енумераторів.

b) Тепер створіть клас Monster з наступними трьома атрибутами (змінними-членами): тип (MonsterType), ім’я (std::string) і кількість здоров’я (int).

c) Перерахування MonsterType є специфічним для Monster, тому перемістіть його всередину класу під специфікатор доступу public.

d) Створіть конструктор, який дозволить ініціалізувати всі змінні-члени класу.

Наступний фрагмент коду повинен скомпілюватися без помилок:

```
int main()
{
	Monster jack(Monster::Orc, "Jack", 90);

	return 0;
}
```

e) Тепер нам потрібно вивести інформацію про нашого монстра. Для цього потрібно конвертувати MonsterType в std::string. Додайте функцію getTypeString(), яка виконуватиме конвертацію, і функцію виводу print().

Наступна програма:

```
int main()
{
	Monster jack(Monster::Orc, "Jack", 90);
	jack.print();

	return 0;
}
```

Повинна видавати наступний результат:

Jack is the orc that has 90 health points.

f) Тепер ми вже можемо створити сам генератор монстрів. Для цього створіть статичний клас MonsterGenerator і статичний метод з ім'ям generateMonster(), який повертатиме випадкового монстра. Поки що метод нехай повертає анонімний об'єкт: (Monster::Orc, "Jack", 90).

Наступна програма:

```
int main()
{
	Monster m = MonsterGenerator::generateMonster();
	m.print();

	return 0;
}
```

Повинна видавати наступний результат:

Jack is the orc that has 90 health points.

g) Тепер MonsterGenerator повинен генерувати деякі випадкові атрибути. Для цього нам знадобиться генератор випадкового числа. Скористайтеся наступною функцією:

```
int main()
{
	Monster m = MonsterGenerator::generateMonster();
	m.print();

	return 0;
}
```

Оскільки MonsterGenerator покладатиметься безпосередньо на цю функцію, то помістіть її всередину класу в якості статичного методу.

h) Тепер змініть функцію generateMonster() для генерації випадкового MonsterType (між 0 і Monster::MAX_MONSTER_TYPES-1) і випадкової кількості здоров'я (від 1 до 100). Це має бути просто. Після того, як ви це зробите, визначте один статичний фіксований масив s_names розміром 6 елементів всередині функції generateMonster() і ініціалізуйте його шістьма будь-якими іменами на ваш вибір. Додайте можливість вибору випадкового імені з цього масиву.

Наступний фрагмент коду повинен скомпілюватися без помилок:

```
#include <ctime> // для time()
#include <cstdlib> // для rand() і srand()

int main()
{
```

```
	srand(static_cast<unsigned int>(time(0))); // використовуємо системний годинник в якості стартового значення
	rand(); // користувачам Visual Studio: скидаємо перше згенероване (рандомне) число

	Monster m = MonsterGenerator::generateMonster();
	m.print();

	return 0;
}
```