Download the starter code `main.cpp`, `Character.h` and `Character.cpp` from CCLE. Do not modify `Character.h`. We have provided `main.cpp` to give you an idea of how we intend to use the class. Put the implementations into `Character.cpp`. You may not use global variables. Submit **only** `Character.cpp` to CCLE.

Problem: (Combat game)

Write the implementations for the class `Character`, which simulates a text combat game, into the file `Character.cpp`.

The private data members:

- `string name` represents the name of the character;
- `int health` represents the health of the character;
- `int damage` represents the damage to a target's health after the character uses a regular attack;
- `int arrows` represents the number of arrows the character has.

A constructor with default values for `health`, `damage` and `arrows` has been included in `Character.cpp`.

The other constructor `Character(string newName, int newHealth, int newDamage, int newArrows)` sets the name, health, damage, arrows of a new character.

The public methods:

- `void set_health(int newHealth)` sets the health of the character;
- `string get_name() const` returns the name of the character;
- `int get_health() const` returns the health of the character;
- `void display() const` prints the health of the character and the number of arrows the character has to the console;
  A sample print may look like

  ```
  Hero  health: 44  arrows: 5
  ```

Two types of attacks will be supported: regular attack and ranged attack.

- The public method `void attack(Character& target)` simulates a regular attack to the target. The amount of damage to the target's health caused by the attack is determined by the value of the `damage` data member of the character. It then prints out the result of the attack and the health of the target. A sample run of `Monster` attacking `Hero` may look like

  ```
  Monster attacks Hero doing 3 damage!
  Hero health: 17
  ```

- The public method `void rangedAttack(Character& target)` simulates an arrow attack to the target, and the amount of damage caused by the attack is determined by a random integer between 1 and 5. Every arrow attack costs one arrow. If the character has no arrows left, print something like

  ```
  Hero is out of arrows!
  ```

  to the console. Otherwise a sample run of an arrow attack may look like

  ```
  Hero shoots Monster doing 1 damage!
  Monster health: 9
  ```

Eventually a sample run of `main.cpp` using the class may look like:

```
Monster attacks Hero doing 3 damage!
Hero health: 17
----------------------
Hero  health: 17  arrows: 2
What do you do? 1 attack, 2 fire arrow, Q exit: 2
Hero shoots Monster doing 4 damage!
Monster health: 6
Monster attacks Hero doing 3 damage!
Hero health: 14
----------------------
Hero  health: 14  arrows: 1
What do you do? 1 attack, 2 fire arrow, Q exit: 2
Hero shoots Monster doing 1 damage!
Monster health: 5
Monster attacks Hero doing 3 damage!
Hero health: 11
----------------------
Hero  health: 11  arrows: 0
What do you do? 1 attack, 2 fire arrow, Q exit: 2
Hero is out of arrows!
Monster attacks Hero doing 3 damage!
Hero health: 8
----------------------
Hero  health: 8  arrows: 0
What do you do? 1 attack, 2 fire arrow, Q exit: 1
Hero attacks Monster doing 2 damage!
Monster health: 3
Monster attacks Hero doing 3 damage!
Hero health: 5
----------------------
Hero  health: 5  arrows: 0
```

```
What do you do? 1 attack, 2 fire arrow, Q exit: 1
Hero attacks Monster doing 2 damage!
Monster health: 1
Monster attacks Hero doing 3 damage!
Hero health: 2
----------------------
Hero  health: 2  arrows: 0
What do you do? 1 attack, 2 fire arrow, Q exit: 1
Hero attacks Monster doing 2 damage!
Monster health: -1
Congratulations! You killed the monster!
Thanks for playing!
```