O objetivo deste exercício é avançar no uso de Orientação a Objetos e praticar a parte básica de Herança.

Você deve fazer um programa para administrar um campeonato de batalhas entre Pokemons. Para isso, deverá implementar diferentes classes de acordo com as especificações abaixo:

Pokemon:
Atributos: string _nome; string _tipo_ataque; double _forca_ataque; double _energia = 100;
Pokemon(string nome, string tipo_ataque, double forca_ataque): método construtor para a inicialização dos atributos.
void falar_nome(): imprime o nome do pokemon no formato: "Nome!" (seguido de quebra de linha).
void falar_tipo_ataque(): imprime o tipo do ataque no formato: "Tipo!" (seguido de quebra de linha).
void imprimir_status(): imprime o estado atual do pokemon no formato:

Nome!
Energia: Y

void imprimir_informacoes(): imprime as informações do pokemon no formato:

Pokemon: NOME
Tipo ataque: TIPO
Dano: X
Energia: Y

virtual double calcular_dano() = 0: método virtual puro que calcula o dano de um ataque de acordo com o tipo e dados específicos de cada pokemon.
void atacar(Pokemon* outro_pokemon): método que executa o ataque sobre outro pokemon de acordo com a seguinte lógica:
- Fala o próprio nome
- Fala o tipo do ataque

- Calcula o dano e imprime: "Dano: X"
- Aplica o dano no outro Pokemon

void receber_dano(double valor_dano): cálculo que aplica o dano recebido por um ataque. O valor do dano informado deve ser subtraído da energia atual do pokemon. A energia nunca deve ficar negativa, sendo o menor valor válido 0. Caso a energia seja zerada, deve-se imprimir a mensagem: "NOME morreu!" (seguido de quebra de linha).

PokemonEletrico (herda de Pokemon):
Atributos: double _potencia_raio;
PokemonEletrico(string nome, string tipo_ataque, double forca_ataque, double potencia_raio): método construtor para a inicialização dos atributos.
void falar_tipo_ataque(): sobrescrita do método da superclasse. Além de escrever o tipo do ataque também escreve "Bzzzz!" (seguido de quebra de linha).
double calcular_dano(): sobrescrita do método da superclasse. Retorna o valor a partir do ataque específico.
double ataque_eletrico(): calcula e retorna o valor do dano do ataque específico. Calculado da seguinte forma: forca_ataque * potencia_raio.

PokemonAquatico (herda de Pokemon):
Atributos: double _litros_jato;
PokemonAquatico(string nome, string tipo_ataque, double forca_ataque, double litros_jato): método construtor para a inicialização dos atributos.
void falar_tipo_ataque(): sobrescrita do método da superclasse. Além de escrever o tipo do ataque também escreve "Splash!" (seguido de quebra de linha).
double calcular_dano(): sobrescrita do método da superclasse. Retorna o valor a partir do ataque específico.
double ataque_aquatico(): calcula e retorna o valor do dano do ataque específico. Calculado da seguinte forma: forca_ataque + litros_jato.

PokemonExplosivo (herda de Pokemon):
Atributos: double _temperatura_explosao;
PokemonExplosivo(std::string nome, std::string tipo_ataque, double forca_ataque, double temperatura_explosao): método construtor para a inicialização dos atributos.

void falar_tipo_ataque(): sobrescrita do método da superclasse. Além de escrever o tipo do ataque também escreve "Boom!" (seguido de quebra de linha).
double calcular_dano(): sobrescrita do método da superclasse. Retorna o valor a partir do ataque específico.
double ataque_explosivo(): calcula e retorna o valor do dano do ataque específico. Calculado da seguinte forma: forca_ataque / temperatura_explosao.

Treinador:
Atributos: string _nome; vector<Pokemon*> _pokemons;
Métodos para cadastro de pokemons específicos (devem ser inseridos sequencialmente no vetor):
void cadastrar_pokemon_eletrico(string nome, string tipo_ataque, double forca_ataque, double potencia_raio)
void cadastrar_pokemon_aquatico(string nome, string tipo_ataque, double forca_ataque, double litros_jato)
void cadastrar_pokemon_explosivo(string nome, string tipo_ataque, double forca_ataque, double temperatura_explosao)
Pokemon* usar_pokemon(int idpk): retorna um pokemon específico a partir do índice informado.
void imprimir_informacoes(): imprime as informações do treinador no seguinte formato:

- Se nenhum Pokemon foi cadastrado

Nome: NOME

Nenhum Pokemon cadastrado!

- Caso contrário imprime as informações dos Pokemons

Nome: NOME

(Informações Pokemon 0)

(Informações Pokemon 1)

(etc)

Campeonato:
Atributos: vector<Treinador*> _treinadores;
void cadastrar_treinador(std::string nome): método para cadastro de treinadores (devem ser inseridos sequencialmente no vetor).
Métodos para cadastro de pokemons específicos para um determinado treinador (de acordo com o índice):
void cadastrar_pokemon_eletrico(int idt,string nome, string tipo_ataque, double forca_ataque, double potencia_raio)
void cadastrar_pokemon_aquatico(int idt, string nome, string tipo_ataque, double forca_ataque, double litros_jato)
void cadastrar_pokemon_explosivo(int idt, string nome, string tipo_ataque, double forca_ataque, double temperatura_explosao)
void imprimir_informacoes_treinador(int idt): método que imprime as informações de um determinado treinador (de acordo com o índice).
void executar_batalha(int idt1, int idpk1, int idt2, int idpk2): executa uma batalha considerandos os Treinadores/Pokemons informados. A batalha imprime as seguintes informações:

Batalha
TREINADOR1 (POKEMON1) vs. TREINADOR2 (POKENON2)
- Chamar método atacar (nesse caso, POKEMON1 ataca o POKEMON2)
- Imprimir o status do pokemon atacado (nesse caso, POKEMON2)
#########

Atenção, todos os atributos devem ser privados e acessados/manipulados fora das classes apenas através de métodos auxiliares. Você é livre para adicionar quaisquer outros atributos ou métodos que julgar necessário. Além disso, lembre-se de fazer a

correta manipulação da memória ao utilizar ponteiros (o destrutor é um bom lugar para isso!). Faça também a correta modularização utilizando os arquivos .hpp e .cpp.

Por fim, você deve implementar o arquivo main.cpp e adicionar toda a parte de entrada/saída que será responsável por manipular os seguintes comandos:

't nome': comando para cadastrar um treinador no campeonato.
'i idt': comando para imprimir as informações de um treinador de acordo com o id.
'e idt nome tipo_ataque forca_ataque potencia_raio': comando para adicionar um PokemonEletrico ao treinador informado.
'q idt nome tipo_ataque forca_ataque litros_jato': comando para adicionar um PokemonAquatico ao treinador informado.
'x idt nome tipo_ataque forca_ataque temperatura_explosao_jato': comando para adicionar um PokemonExplosivo ao treinador informado.
'h idt1 idpk1 idt1 idpk1': comando para realizar uma batalha de acordo com os ids informados.
'b': deve chamar a função 'avaliacao_basica()' implementada no arquivo "avaliacao_basica_pokemon.hpp" (já incluído no main.cpp). Essa função faz uma avaliação complementar do código (não apenas dos resultados).
Você pode assumir que todas as strings que serão informadas *não* possuem espaço, ou seja, são palavras únicas.

Para ilustrar, abaixo é apresentado um exemplo de entrada/saída:

input=
t Treinador0
i 0
e 0 Pikachu Relampago 5 2
i 0
t Treinador1
q 1 Squirtle Tsunami 5 3
i 1
h 0 0 1 0
h 1 0 0 0

i 0

i 1

output=

Nome: Treinador0

Nenhum Pokemon cadastrado!

Nome: Treinador0

Pokemon: Pikachu

Tipo ataque: Relampago

Dano: 10

Energia: 100

Nome: Treinador1

Pokemon: Squirtle

Tipo ataque: Tsunami

Dano: 8

Energia: 100

Batalha

Treinador0 (Pikachu) vs. Treinador1 (Squirtle)

Pikachu!

Relampago!

Bzzzz!

Dano: 10

Squirtle!

Energia: 90

#########

Batalha

Treinador1 (Squirtle) vs. Treinador0 (Pikachu)

Squirtle!

Tsunami!
Splash!
Dano: 8
Pikachu!
Energia: 92
#########
Nome: Treinador0

Pokemon: Pikachu
Tipo ataque: Relampago
Dano: 10
Energia: 92

Nome: Treinador1

Pokemon: Squirtle
Tipo ataque: Tsunami
Dano: 8
Energia: 90

Dica 1:
O código da avaliação básica pode ser copiado aqui, caso você queira depurar algo localmente.

Dica 2:
Você pode usar os códigos dos exercícios anteriores e o da avaliação básica para lhe ajudar a fazer toda a parte de entrada/saída.

Referências:
https://www.cplusplus.com/reference/stl/
https://www.cplusplus.com/doc/tutorial/classes/
https://www.cplusplus.com/doc/tutorial/inheritance/
https://www.cplusplus.com/reference/iomanip/setprecision/