HW5

HW5

- 클래스 이름: Person
  - Data:
    - 나이 (integer)
    - 이름 (string)
    - 성별 (char), F=여성, M=남성
    - 학교이름 (string)
    - 좋아하는 것들 (vector<string>): 좋아하는것들의 리스트. 중복 불가
    - 싫어하는 것들 (vector<string>): 싫어하는것들의 리스트. 중복 불가
    - 단, 같은 내용이 좋아하는 것과 싫어하는것에 중복되어 있을 수 없다

Sol

```
Person(string name, char gender) {
      this->name = name;
      age = 1;

      if(gender != 'M' && gender != 'F')
         this->gender = 'F';
      else
         this->gender = gender;

      schoolname = "NO SCHOOL";
      favorite.clear();
      dislike.clear();
   }
```

HW5

- Getters
  - string get_name()
  - int get_age()
  - char get_gender()
  - string get_favorite(int i): i번째 좋아하는것
  - string get_dislike(int i): i번째 싫어하는것
  - int get_fav_cnt(): 좋아하는것의 갯수
  - int get_dis_cnt(): 싫어하는것의 갯수

Sol

```
string Person::get_name() const {
    return this->name;
}
int Person::get_age() const {
    return this->age;
}
char Person::get_gender() const {
    return this->gender;
}
string Person::get_favorite(int i) const {
    return this->favorite.at(i-1);
}
```

```
string Person::get_dislike(int i) const {
    return this->dislike.at(i-1);
}
int Person::get_fav_cnt() const {
    return this->favorite.size();
}
int Person::get_dis_cnt() const {
    return this->dislike.size();
}
```

HW5

- Setters
  - void set_name(string name)
  - void set_age(int age)

Sol

```
void Person::set_name(string name) {
    this->name = name;
}
void Person::set_age(int age) {
    this->age = age;
}
```

HW5

- Checkers
  - bool is_female(): 여자면 참, 아니면 거짓
  - bool is_schoolkid(): 나이가 7살 이상, 19살 이하면 참

Sol

- bool Person::is_female() {
- return this->gender == 'F';
- }

- bool Person::is_schoolkid() {
- return this->age >= 7 && this->age <= 19;
- }

HW5

- Operator overloading
  - operator++(prefix): 나이 1 증가
  - operator>: 나이를 비교

Sol

```
Person& Person::operator++() {
    age++;
    return *this;
}

bool operator >(const Person lhs, const Person rhs)
{
    return lhs.age > rhs.age;
}
```

HW5

- Other functions
  - void goto_school(string schoolname)
    - 학교이름을 설정해준다. 단, 나이가 7살이상 19살 이하가 아니면 "NO SCHOOL"로
  - void add_favorite(string thing)
    - thing을 좋아하는것들에 추가 (새로운 것만)
    - 싫어하는 것들에 있으면 삭제
  - void add_dislike(string thing)
    - thing을 싫어하는 것들에 추가 (새로운 것만)
    - 좋아하는 것들에 있으면 삭제

Sol

```
void Person::goto_school(string schoolname) {
    if(!is_schoolkid())
        this->schoolname = schoolname;
    else
        this->schoolname = "NO SCHOOL";
}
void Person::add_favorite(string thing) {
   // add thing to favorite
   // remove from dislike
}
void Person::add_dislike(string thing) {
   // add thing to dislike
   // remove from favorite
}
```

Sol

```
void Person::add_favorite(string thing) {
      vector<string>::iterator it;

      // add thing to favorite
       if( find( fav.begin(), fav.end(), thing ) == fav.end() )
            fav.push_back(thing);

      // remove from dislike
       if( (it=find( dislike.begin(), dislike.end(), thing )) != fav.end() )
            dislike.erase(it);
}
```

Test Code

```
int main() {
   Person p1("철수", 'M');
   check("1. 이름", p1.get_name(), "철수"); // 생성자

   Person p2("영희", 'S');
   check("2. 성별", p2.get_gender(), 'F'); // 생성자

   Person p3("태희", 'F');
   p3.set_name("혜교");
   check("3. 이름", p3.get_name(), "혜교"); // getter 점검
     p3.set_age(15);
   check("4. 나이", p3.is_schoolkid(), true); // checker 점검

   Person p4("영수", 'M');
   Person p5("민아", 'T');
   p4++;
   check("5. 나이", p4 < p5, false); // opertor overloading 점검

 Person p7("민식", 'M');
   p7.add_favorite("자동차");
   p7.add_favorite("영화");
   p7.add_favorite("책");
   p7.add_favorite("장난감");
   p7.add_favorite("장난감");
   p7.add_favorite("영화");
   check("6. 추가", p7.get_favorite(2), "책"); // vector 중복점검 (2점)
    check("7. 추가", p7.get_fav_cnt(), 4); // vector 갯수 점검

   Person p8("강현", 'F');
   p8.add_favorite("노래");
   p8.add_favorite("춤");
   p8.add_favorite("가수");
   p8.add_dislike("춤");
   check("8. 추가", p8.get_favorite(2), "가수"); // vector 중복제거 점검 (2점)

   system("pause");
}
```

Advanced Sol for add_fav/dislike

- algorithm::remove(begin, end, val)
  - move all matching items to the end of the list
  - return first position of moved items
    - //x contains (dog, cat, dog, cow)
    - it = remove(x.begin(), x.end(), “dog”)
    - //x conains (cat, cow, dog, dog)
    - //it points to 3rd item---^
- erase(it)
  - erase at it
- erase(begin, end)
  - erase all from start to end

Advanced Sol for add_fav/dislike

it = list.erase(remove(list.begin(), list.end(), thing) , list.end())

- list에서 thing을 모두 찾아서 지운다
- 예
  - list:
    - (dog, cat, cow, dog, duck, bird)
  - after remove
    - (cat, cow, duck, bird, dog, dog)
    - it ----------^
  - after remove erase(it, end)
    - (cat, cow, duck, bird)

Advanced Sol for add_fav/dislike

```
void Person::add_favorite(string thing) {
	this->favorite.erase(
		remove(this->favorite.begin(), this->favorite.end(), thing),
		this->favorite.end()
	);
	this->dislike.erase(remove(this->dislike.begin(), this->dislike.end(), thing), this->dislike.end());
	this->favorite.push_back(thing);
}
void Person::add_dislike(string thing) {
	this->favorite.erase(remove(this->favorite.begin(), this->favorite.end(), thing), this->favorite.end());
	this->dislike.erase(remove(this->dislike.begin(), this->dislike.end(), thing), this->dislike.end());
	this->dislike.push_back(thing);
}
```