Q1: No, it's not a member function, so it cannot be overridden in a derived class.

Q2: When writing 5 + obj, 5 is the first argument. In case of obj + 5, obj becomes the calling object and 5 is sent as an argument. For the statement 5 + obj, 5 would be expected to be the calling object. But 5 is a primitive (int) and cannot be the calling object. So, it would fail.

Q3: Because static members are created regardless of whether there are any objects created or not. Each object is created with its own data members however the static member is not reallocated because it has already been created at the time of program execution.

Q4: No, we cannot tell until we see if performTask() is a virtual function in the parent class

Q5: Yes, it will create ambiguity as the statement will theoretically result in both function calls, so neither can be called / will result in a compile time error.

Part a:

```
class AIComponent {
public:
    virtual void performTask() const = 0;
    virtual ~AIComponent() {}
};
class NavigationAI : virtual public AIComponent {
protected:
    string start, destination; int estimated_time;
public:
    NavigationAI(string s = "", string d = "", int time = 0)
        : start(s), destination(d), estimated_time(time) {}
    void performTask() const override {
        cout << "NavigationAI: Calculating best route...\n";}
    string getStart() const { return start; }
    string getDestination() const { return destination; }};
```

Part b:

```
virtual void NavigationAI::routeMap(string s, string d) {
        start = s; destination = d;
        cout << "Starting navigation from " << start << " to " << destination << ".
Drive safely!\n";}


class EntertainmentAI : virtual public AIComponent {
```

```
    int volume;
    string playlist[10]; // Fixed size playlist
    int playlistCount;
public:
    EntertainmentAI(int vol = 5) : volume(vol), playlistCount(0) {}
    void addToPlaylist(const string& song) {
        if (playlistCount < 10) {
            playlist[playlistCount++] = song;}}
    void performTask() const override {
        cout << "EntertainmentAI: Playing music and responding to voice
commands...\n";}
    void printPlaylist() const {
        cout << "Playlist:\n";
        for (int i = 0; i < playlistCount; ++i) {
            cout << " - " << playlist[i] << '\n';
        }}};
class SelfDrivingAI : public NavigationAI, public EntertainmentAI {
    int current_speed;
public:
    SelfDrivingAI(string s = "", string d = "", int time = 0, int speed = 0)
        : NavigationAI(s, d, time), current_speed(speed) {}
    void performTask() const override {
        cout << "SelfDrivingAI: Making real-time autonomous driving decisions...\n";}
    void routeMap(string s, string d) override {
        cout << "SelfDrivingAI: Asking for entertainment preferences...\n";
        NavigationAI::routeMap(s, d);}};
```

Part c:

```
void demonstratePolymorphism() {
    AIComponent* modules[3];
    modules[0] = new NavigationAI("Home", "Work", 15);
    modules[1] = new EntertainmentAI();
    modules[2] = new SelfDrivingAI("Home", "Work", 15, 50);
    for (int i = 0; i < 3; i++) {
```

```
        modules[i]->performTask();
        delete modules[i];
    }}
```

Part d:

```
//in NavigationAI class
friend bool compareRoutes(const NavigationAI&, const NavigationAI&);
//globa bool compareRoutes(const NavigationAI& a, const NavigationAI& b) {
    return (a.start == b.start && a.destination == b.destination);
}l
```

Part e:

```
//goes in SelfDrivingAI class
friend class RemoteAIControl;
//global
class RemoteAIControl {
public:
    void changeSpeed(SelfDrivingAI& car, int newSpeed) {
        car.current_speed = newSpeed;
        cout << "RemoteAIControl: Speed updated to " << newSpeed << " km/h\n";}
    void changeDestination(SelfDrivingAI& car, const string& newDest) {
        car.destination = newDest;
        cout << "RemoteAIControl: Destination changed to " << newDest << "\n";
    }};
```

Part f:

```
//in NavigationAI class
void NavigationAI::operator*(int multiplier) {
        estimated_time *= multiplier;}
//in SelfDrivingAI class
bool SelfDrivingAI::operator==(const NavigationAI& nav) {
        return (start == nav.getStart() && destination == nav.getDestination());}
//in Entertainment AI class
void EntertainmentAI::operator+=(const EntertainmentAI& other) {
        for (int i = 0; i < other.playlistCount && playlistCount < 10; ++i) {
            playlist[playlistCount++] = other.playlist[i];}}
```