ARMADA TEST CASES

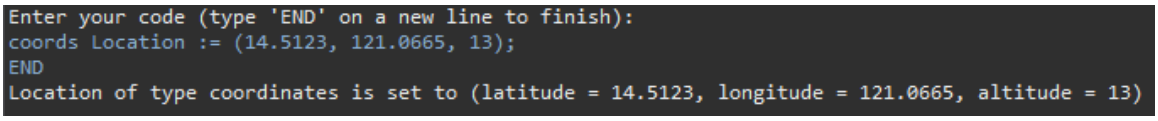
1. **coord (Main.java)**

Valid Input:

```
coords Location := (14.5123, 121.0665, 13);
END
```

Expected Output:

```
Location of type coordinates is set to (latitude = 14.5123, longitude = 121.0665, altitude = 13)
```

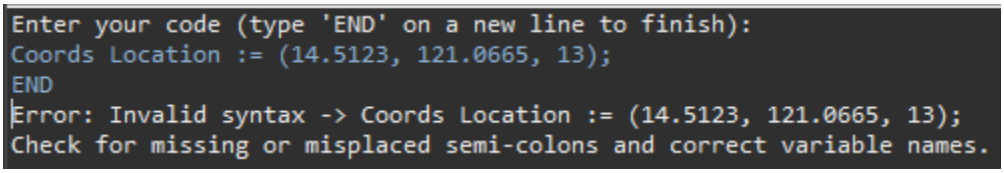


Invalid Input:

```
Coords Location := (14.5123, 121.0665, 13);
END
```

Expected Output:

```
Error: Invalid syntax -> Coords Location := (14.5123, 121.0665, 13);
Check for missing or misplaced semi-colons and correct variable names.
```

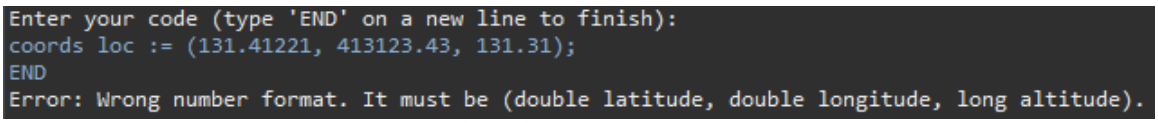


Invalid Input:

```
coords loc := (131.41221, 413123.43, 131.31);
END
```

Expected Output:

```
Error: Wrong number format. It must be (double latitude, double longitude, long altitude).
```

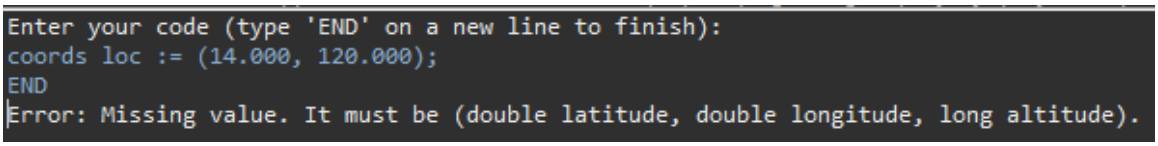


Invalid Input:

```
coords loc := (14.000, 120.000);
END
```

Expected Output:

```
Error: Missing value. It must be (double latitude, double longitude, long altitude).
```



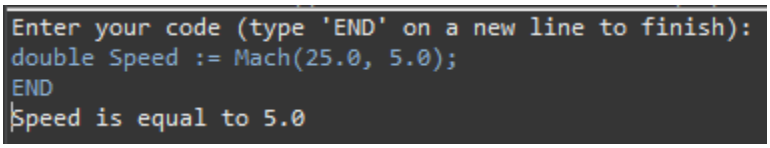
2. **Mach (Main.java)**

Valid Input:

```
double Speed := Mach(25.0, 5.0);
END
```

Expected Output:

```
Speed is equal to 5.0
```

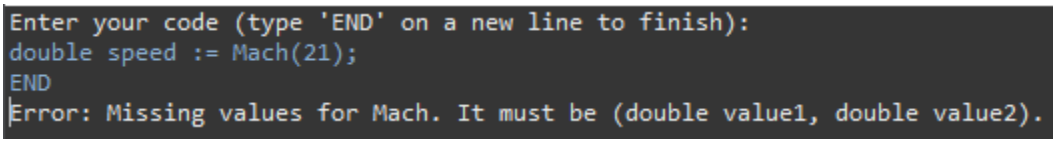


Invalid Input:

```
double speed := Mach(21);
END
```

Expected Output:

```
Error: Missing values for Mach. It must be (double value1, double value2).
```



Invalid Input:

```
long Speed := Mach(25.0, 5.0);
END
```

Expected Output:

```
Error: Invalid syntax -> long Speed := Mach(25.0, 5.0);
Check for missing or misplaced semi-colons and correct variable names.
```

```
Enter your code (type 'END' on a new line to finish):
long Speed := Mach(25.0, 5.0);
END
Error: Invalid syntax -> long Speed := Mach(25.0, 5.0);
Check for missing or misplaced semi-colons and correct variable names.
```

3. **print (Main.java)**

Valid Input:

```
print("Hello World!");
END
```

Expected Output:

```
Hello World!
```

```
Enter your code (type 'END' on a new line to finish):
print("Hello World!");
END
Hello World!
```

Valid Input:

```
coords Location := (14.5123, 121.0665, 13);
print(Location);
END
```

Expected Output:

```
Location of type coordinates is set to (latitude = 14.5123, longitude = 121.0665, altitude = 13)
(latitude = 14.5123, longitude = 121.0665, altitude = 13)
```

```
Enter your code (type 'END' on a new line to finish):
coords Location := (14.5123, 121.0665, 13);
print(Location);
END
Location of type coordinates is set to (latitude = 14.5123, longitude = 121.0665, altitude = 13)
(latitude = 14.5123, longitude = 121.0665, altitude = 13)
```

Invalid Input:

```
Print(“Hello World!”);
END
```

Expected Output:

```
Error: Invalid syntax -> Print(“Hello World!”);
Check for missing or misplaced semi-colons and correct variable names.
```

```
Enter your code (type 'END' on a new line to finish):
Print(“Hello World!”);
END
Error: Invalid syntax -> Print(“Hello World!”);
Check for missing or misplaced semi-colons and correct variable names.
```

Invalid Input:

```
print(location);
END
```

Expected Output:

```
Error: Identifier 'location' not found. Check variable declarations.
```

```
Enter your code (type 'END' on a new line to finish):
print(location);
END
Error: Identifier 'location' not found. Check variable declarations.
```

Invalid Input:

```
print (“Hello World!”);
END
```

Expected Output:

```
Error: Invalid syntax -> print (“Hello World!”);
Check for missing or misplaced semi-colons and correct variable names.
```

```
Enter your code (type 'END' on a new line to finish):
print (“Hello World!”);
END
Error: Invalid syntax -> print (“Hello World!”);
Check for missing or misplaced semi-colons and correct variable names.
```

4. **case (IfCFG.java)**

Valid Input:

```
case (a == b) {
   // statements
}
case (x > 10) {
   // statements
}
case (value >= 20) {
   // statements
}
END
```

Expected Output:

```
Valid case statement: case (a == b) {
Valid case statement: case (x > 10) {
Valid case statement: case (value >= 20) {
```

```
Enter your code (type 'END' on a new line to finish):
case (a == b) {
        // statements
}
case (x > 10) {
        // statements
}
case (value >= 20) {
        // statements
}
END
Valid case statement: case (a == b) {
Valid case statement: case (x > 10) {
Valid case statement: case (value >= 20) {
```

Valid Input:

```
status flightStatus;
coords location := (0, 0, 0);
coords NAIA := (14.5123, 121.0165, 13);
coords DIA := (7.122552, 125.64550, 22);

case (location == NAIA) {
   flightStatus := "Landed";
}
case (location != NAIA && location != DIA) {
   flightStatus := "Airborne";
}
case (location == DIA) {
   flightStatus := "Boarding";
}
END
```

Expected Output:

```
Data type declared flightStatus of type status.
location of type coordinates is set to (latitude = 0.0, longitude = 0.0,
altitude = 0)
NAIA of type coordinates is set to (latitude = 14.5123, longitude =
121.0165, altitude = 13)
DIA of type coordinates is set to (latitude = 7.122552, longitude =
125.6455, altitude = 22)
```

```
Valid case statement: case (location == NAIA) {
flightStatus of type status is set to Landed.
Valid case statement: case (location != NAIA && location != DIA) {
flightStatus of type status is set to Airborne.
Valid case statement: case (location == DIA) {
flightStatus of type status is set to Boarding.
```

```
Enter your code (type 'END' on a new line to finish):
status flightStatus;
coords location := (0, 0, 0);
coords NAIA := (14.5123, 121.0165, 13);
coords DIA := (7.122552, 125.64550, 22);

case (location == NAIA) {
        flightStatus := "Landed";
}
case (location != NAIA && location != DIA) {
        flightStatus := "Airborne";
}
case (location == DIA) {
        flightStatus := "Boarding";
}
END
Data type declared flightStatus of type status.
location of type coordinates is set to (latitude = 0.0, longitude = 0.0, altitude = 0)
NAIA of type coordinates is set to (latitude = 14.5123, longitude = 121.0165, altitude = 13)
DIA of type coordinates is set to (latitude = 7.122552, longitude = 125.6455, altitude = 22)
Valid case statement: case (location == NAIA) {
flightStatus of type status is set to Landed.
Valid case statement: case (location != NAIA && location != DIA) {
flightStatus of type status is set to Airborne.
Valid case statement: case (location == DIA) {
flightStatus of type status is set to Boarding.
```

Invalid Input:

```
case (x != ) {
   // stmts
}
case (temp <) {
   // stmts
}
END
```

Expected Output:

```
Invalid case statement (invalid condition): case (x != ) {
Invalid case statement (invalid condition): case (temp <) {
```

```
Enter your code (type 'END' on a new line to finish):
case (x != ) {
        // stmts
}
case (temp <) {
        // stmts
}
END
Invalid case statement (invalid condition): case (x != ) {
Invalid case statement (invalid condition): case (temp <) {
```

Invalid Input:

```
case (a == b) {
END
```

Expected Output:

```
Invalid case statement (invalid condition): case (x != ) {
Invalid case statement (invalid condition): case (temp <) {
```

```
Enter your code (type 'END' on a new line to finish):
case (a == b) {
END
Invalid case statement (missing closing brace): case (a == b) {
```

Invalid Input:

```
case (location) {
}
END
```

Expected Output:

```
Invalid case statement (invalid condition): case (location) {
```

```
Enter your code (type 'END' on a new line to finish):
case (location) {
}
END
Invalid case statement (invalid condition): case (location) {
```

5. **create (ObjectCFG.java)**

Valid Input:

```
create Object02{
double speed;
status flight_status;
}
END
```

Expected Output:

```
Valid object creation: Object02{
Field added: speed of type double.
Field added: flight_status of type status.
Object creation completed.
```

```
Enter your code (type 'END' on a new line to finish):
create Object02{
   double speed;
   status flight_status;
}
END
Valid object creation: Object02{
Field added: speed of type double.
Field added: flight_status of type status.
Object creation completed.
```

Valid Input:

```
create Object {
   string name;
   coords location;
   double speed;
   status flightStatus;
}
Object airplane;
airplane.name := "Boeing";
airplane.location := (14.5123, 121.0165, 13);
airplane.speed := Mach(25.0, 5.0);
airplane.flightStatus := "Landed";
print(airplane.name);
print(airplane.location);
print(airplane.speed);
END
```

Expected Output:

```
Valid object creation: Object
Field added: name of type string.
Field added: location of type coords.
Field added: speed of type double.
Field added: flightStatus of type status.
Object creation completed.
Object instance created: airplane of type Object
Assigned string to airplane.name
Assigned coords to airplane.location
Assigned Mach result to airplane.speed
```

```
Assigned status to airplane.flightStatus
Field value for airplane.name: Boeing
Field value for airplane.location: Coordinates (latitude=14.5123,
longitude=121.0165, altitude=13)
Field value for airplane.speed: 125.0
```

```
Enter your code (type 'END' on a new line to finish):
create Object {
        string name;
        coords location;
        double speed;
        status flightStatus;
}
Object airplane;
airplane.name := "Boeing";
airplane.location := (14.5123, 121.0165, 13);
airplane.speed := Mach(25.0, 5.0);
airplane.flightStatus := "Landed";
print(airplane.name);
print(airplane.location);
print(airplane.speed);
END
Valid object creation: Object
Field added: name of type string.
Field added: location of type coords.
Field added: speed of type double.
Field added: flightStatus of type status.
Object creation completed.
Object instance created: airplane of type Object
Assigned string to airplane.name
Assigned coords to airplane.location
Assigned Mach result to airplane.speed
Assigned status to airplane.flightStatus
Field value for airplane.name: Boeing
Field value for airplane.location: Coordinates (latitude=14.5123, longitude=121.0165, altitude=13)
Field value for airplane.speed: 125.0
```

Invalid Input:

```
create Object {
   string name;
   coords location;
   double speed;
   status flightStatus;
END
```

Expected Output:

```
Valid object creation: Object
Field added: name of type string.
Field added: location of type coords.
Field added: speed of type double.
Field added: flightStatus of type status.
Error: Object creation incomplete, missing closing brace.
```

```
Enter your code (type 'END' on a new line to finish):
create Object {
        string name;
        coords location;
        double speed;
        status flightStatus;
END
Valid object creation: Object
Field added: name of type string.
Field added: location of type coords.
Field added: speed of type double.
Field added: flightStatus of type status.
Error: Object creation incomplete, missing closing brace.
```

Invalid Input:

```
airplane.name := "Boeing";
airplane.location := (14.5123, 121.0165, 13);
airplane.speed := Mach(25.0, 5.0);
airplane.flightStatus := "Landed";
END
```

Expected Output:

```
Error: Object airplane not found.
Error: Object airplane not found.
Error: Object airplane not found.
Error: Object airplane not found.
```

```
Enter your code (type 'END' on a new line to finish):
airplane.name := "Boeing";
airplane.location := (14.5123, 121.0165, 13);
airplane.speed := Mach(25.0, 5.0);
airplane.flightStatus := "Landed";
END
Error: Object airplane not found.
Error: Object airplane not found.
Error: Object airplane not found.
Error: Object airplane not found.
```

Invalid Input:

```
create Object {
   status flightStatus;
}
Object airplane;
airplane.flightStatus := "Delayed";
END
```

Expected Output:

```
Valid object creation: Object
Field added: flightStatus of type status.
Object creation completed.
Object instance created: airplane of type Object
Error: Invalid status -> "Delayed". Valid statuses are 'Landed', 'Airborne', and 'Boarding'.
```