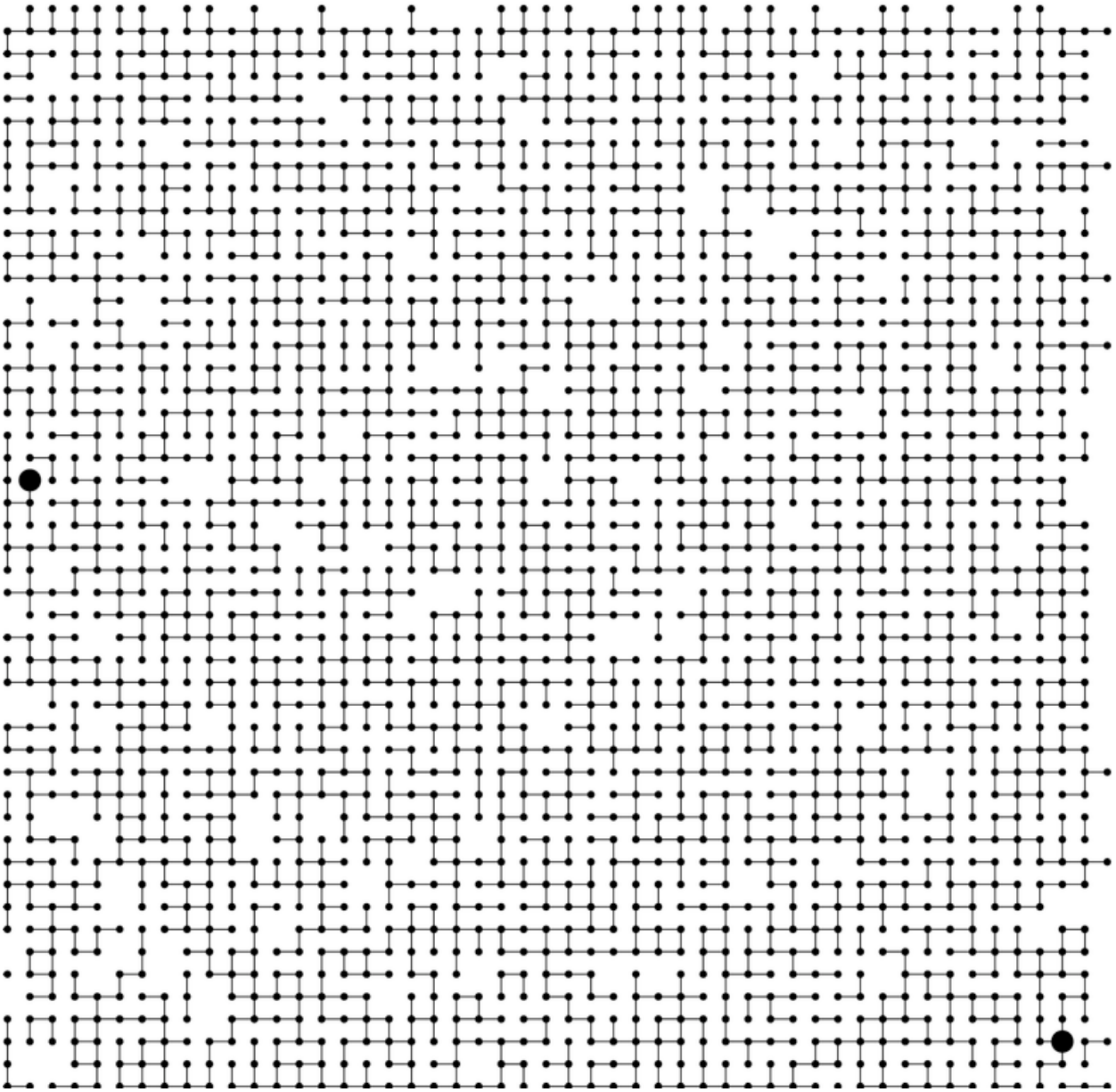
Assignment 1

No program required

Find a path between the two points. Make a copy of page and draw the path on the picture.

Discuss how we could go about designing a computer algorithm to solve the path problem. Don't worry about an exact algorithm. At this point, we only want to think about what issues need to be addressed and what problems we foresee with our algorithm. We will cover these topics during the course later this academic year.



Programming assignments

1. Do problems 1.1.19 and 1.1.39 from textbook.

2. Write a **Person** class that contains the following fields and methods:
   - First Name
   - Last Name
   - A unique ID Number (say, 1001, 1002, etc.). The ID should be assigned by the computer. Use a class variable to keep track of the last ID assigned so that you don't repeat the same number twice.
   - Necessary constructors.
   - Methods to return last name, first name, full name, and ID Number
   - Methods to print last name, first name, and ID Number
   - The **toString** method that returns a neatly formatted string describing the key attributes of the person

   Write a **PersonTeste**r class to test your class.

3. Extend the `Person` class developed above and derive the following classes: **Student**, **HourlyEmployee**, and **FulltimeEmployee**. Determine an appropriate class inheritance hierarchy. These classes should have following fields, necessary constructors, and appropriate access and modifier methods.

   For all employees:
   - Department

   Full-time employees:
   - Salary

   Hourly employees:
   - Hourly rate
   - Number of hours worked each week (4 weeks)

   Student:
   - Classes taken and grades for each class (Use an ArrayList)

   The employee class should contain the necessary methods that will print the total hours (four-week total), average hours per week worked by each employee, and the total wages during a four-week period.

   The student class should contain the necessary methods to print the transcript for each student.

   Write a **Tester** class to test your classes.

4. Write a generic **AddressBook<E extends Person>** class that manages a collection of Person objects. The **AddressBook** class is a generic class limited to person objects. For example, once you instantiate an **AdressBook** of **Student** objects, you cannot add any other type of person objects into that instance of address book.

   The **AddressBook** should have methods to add, delete, or search for a **Person** objects in the address book.

   - The **add** method should add a person object to the address book. Make sure that the **add** method does not add duplicate person objects to the address book.
   - The **delete** method should remove the specified person object from the address book.
   - The **search** method that searches the address book for a specified person and returns the list of persons matching the specified criteria. The search can be done either by first name, last name, or person id.

   Write an **AddressBookTester** class to test your class.

5. Write a **Thermostat** class such that a user of the **Thermostat** class can create an object of **Thermostat** and set it to the desired temperature within a pre-specified range. If the user tries set the temperature outside this range it should throw a **TemperatureTooHigh** or **TemperatureTooLow** exception. Use inheritance to create an exception superclass **TemperatureOutofRange** and subclasses **TemperatureTooHigh** and **TemperatureTooLow**.

   Sample Tester code:

```
Thermostat t = new Thermostat(0, 100);
t. setTemp(50);          // Should be OK.
t.setTemp(150);          // Should throw TemperatureTooHigh  exception.
t.setTemp(-50);          // Should throw TemperatureToolLow exception.
```

   Write a **Tester** class to demonstrate throwing and catching of exceptions. Show that the *catch* specifying the superclass catches the subclass exceptions. The order of exception handlers is important. If you try to catch a superclass exception type before a subclass type, the compiler would generate errors. Also show the re-throwing of exceptions.