San José State University
Engineering Extended Studies

CMPE 202
Software Systems Engineering

Section 47

Spring 2026
Instructor: Ron Mak

Assignment #1

Assigned: Wednesday, January 28
Due: Wednesday, February 4 at 5:30 pm
Individual assignment, 100 points max

Recursion and Backtracking

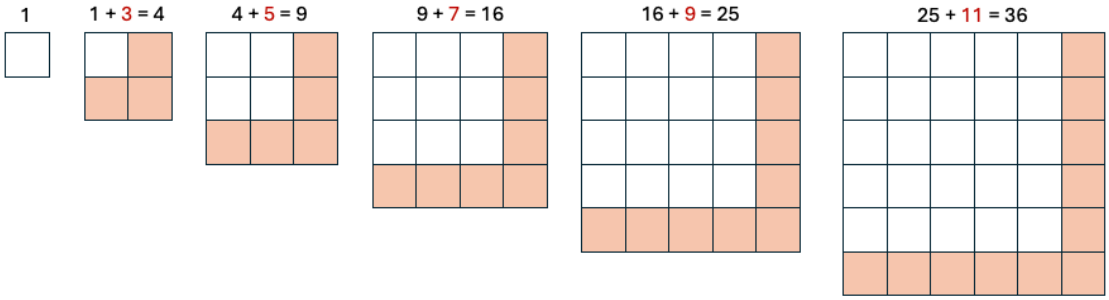
The purpose of the first programming problem is to practice using recursion, and the purpose of the second programming problem is to practice using recursion with backtracking.

Problem 1: Squares of integers with recursion

This table shows a recursive algorithm to compute the squares of the integers 1, 2, 3, ...

n	n^{th} odd	*n* squared
1	1	1
2	3	1 + 3 = 4
3	5	4 + 5 = 9
4	7	9 + 7 = 16
5	9	16 + 9 = 25
6	11	25 + 11 = 36
7	13	36 + 13 = 49
8	15	49 + 15 = 64
9	17	64 + 17 = 81
10	19	81 + 19 = 100

The square of 1 is 1. The square of integer *n*, where *n* > 1, is the square of *n*-1 plus the n^{th} odd number. This diagram shows why this algorithm works:



This algorithm can be proven using induction, the mathematical version of recursion.

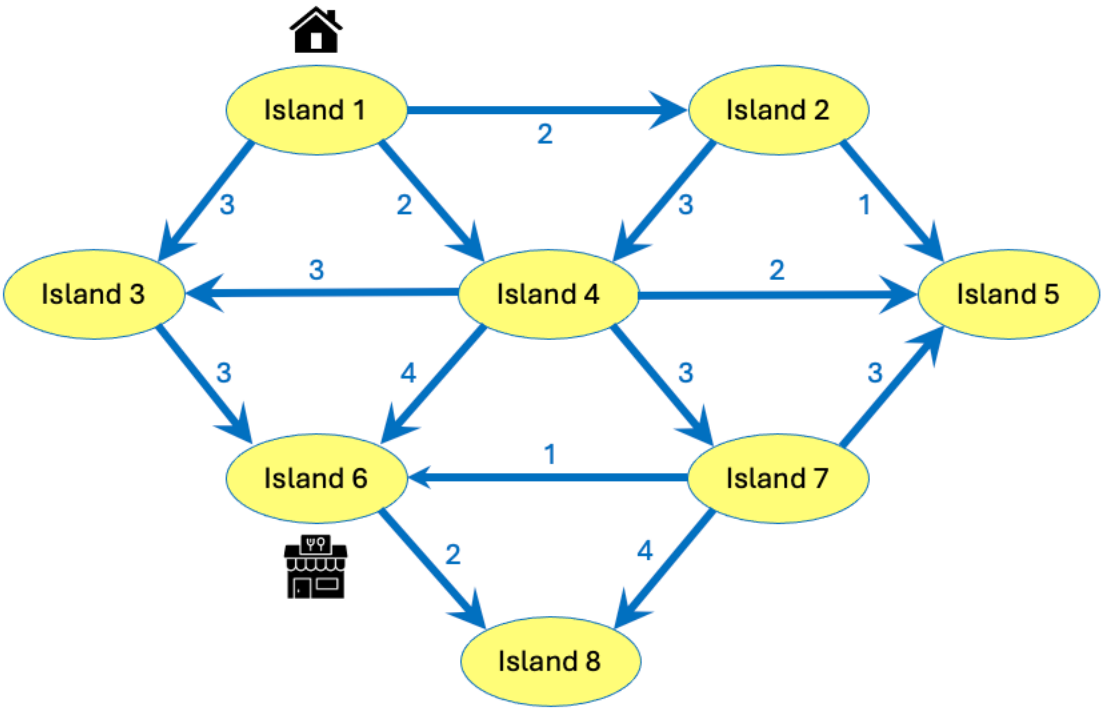
Write a **recursive C++ function** to compute and return the square of an integer *n* passed in as an argument, where *n* >= 1. In your main, use your function to print the squares of the integers from 1 through 10.

In your function's comments, clearly describe:

- What is the base case? What does your function do at the base case?
- What is the simpler case? What does your function do at with this case? How does its action contribute to solving the square of *n*?

Problem 2: Least cost path with recursion and backtracking

You live on Island 1 of a group of eight islands. There is a restaurant on Island 6 that you want to go to from your home. The islands are connected by one-way bridges, and each bridge has a toll.



In the diagram, each arrow represents a bridge and its allowed direction of travel. The number by each bridge is its toll. There are no cycles that will cause you to go in circles.

You are provided with C++ code files **`Island.h`**, **`Bridge.h`**, and **`Graph.h`** that together implement the above graph. Write a program that uses **recursion and backtracking** to compute the lowest cost path from Island 1 to Island 6. There can be several paths that are tied for lowest cost.

Tip: Use recursion to find paths from Island 1 to Island 6. If a path doesn't reach the restaurant at Island 6, backtrack to try another path. After finding a solution path from a given island to Island 6, backtrack to see if there's another solution path from that island.

In your program's comments, clearly describe:

- What is the base case? Is there more than one base case? What does your program do at a base case?
- What is a simpler case? What does your program do at such a case? How does its action contribute to getting to the restaurant at the lowest cost?
- When and how does your program perform backtracking?

Print a path like this: **`Island 1 -> Island 2 -> Island 6`**

Note: Like most recursion-with-backtracking algorithms, this is a brute force way to find a solution. **Dijkstra's algorithm** is an optimal iterative algorithm to find the shortest path or the least cost path. It was developed by famous Dutch computer scientist Edsger Dijkstra (pronounced "dike-stra") in the mid-1950s.

What to turn in

This is an individual assignment. Create two zip files, one containing the code file(s) for Problem 1 and one containing the code files for Problem 2. Include a copy of your program's **runtime output text** in each zip file. Name the zip files after yourself, for example, **`JohnDoe1.zip`** and **`JohnDoe2.zip`**. Then create a combined zip file containing these two zip files. Name this combined zip file after yourself, for example **`JohnDoe.zip`**.

Submit your combined zip file to Canvas under Assignments/Assignment #1.

Rubric

Your submissions will be graded according to these criteria:

Criteria	Maximum points
Problem 1 • Working program • Description of base case actions in comments • Description of recursive actions in comments • Correct output	**10** • 3 • 2 • 2 • 3
Problem 2 • Working program • Description of base case(s) actions in comments • Description of recursive actions in comments • Description of backtracking actions in comments • Correct output	**90** • 20 • 15 • 15 • 15 • 25