1. Area of regular octagon (10 Points)
The area, $A$, of a regular octagon having sides of length $a$ is:

$$A = 2(1 + \sqrt{2})a^2$$

Write a program that calculates and displays the area of a regular octagon. The program will accept the side length as input from the user. The program must display an error message if the length entered is negative.

2. Where’s my car? (10 Points)
You are developing a “Where did I park?” app for smartphones that allows the user to mark where his or her car is parked and then later ask the phone to compute the quickest walking route back to the car. Eventually, the route calculation will be very accurate, but for now you just need to compute a rough approximation of the walking distance to display in a demo version that you are showing to an investor: you’ll just display the distance between the user and the car, neglecting the need to follow streets, sidewalks, etc. Your app knows the latitude ($a_0$) and longitude ($o_0$) where it is currently located and the latitude ($a_1$) and longitude ($o_1$) of where the car is. These are kept in degrees; you must convert them to radians by multiplying by $\pi/180^\circ$ before applying the formula for distance, $d$:

$$d = r \cos(a_0 \cos a_0 \cos a_1 \cos a_1 + \cos a_0 \sin a_0 \cos a_1 \sin a_1 + \sin a_0 \sin a_1)$$

$r$ is the radius of the earth, 3963.1 miles. $\cos$ is the arccosine function. For testing purposes your program will prompt the user for each of the four coordinates in degrees. Longitude normally ranges from -180° to +180°, with positive indicating east of the prime meridian. Latitude normally ranges from -90° to +90° with positive indicating north of the equator. Your program will only be tested with values in these ranges.

To check your program, calculate the distance from the judge’s office (latitude and longitude = 43.043° and -87.905°) to where his car is parked (latitude and longitude = 43.045° and -87.918°). The correct answer is 0.672 miles.

3. Days between dates (10 Points)
Ask the user to input two dates (3 numbers for each indicating, in order, month, date, and year) and then calculate and display the number of days between them. For example, the number of days between November 20 and November 22 in the same year is 2. The number of days between January 3, 2002 and March 6, 2003 = 365 + 31 + 28 + (6-3) = 427. If the user enters an out of range number, display an error message instead of performing the calculation. You may assume that any input from the user is an integer. A year may be any integer. The month must be a number between 1 and 12. The date must be between 1 and the number of days in the month. Assume that February always has 28 days.
4. Area under a curve / Riemann sum (20 Points)

The area under a curve can be estimated by dividing the region under the curve into a sufficiently large number of narrow rectangles and adding up the area of the rectangles. This technique is shown in the figure\(^1\) below and is called numerical integration, specifically a Riemann sum.

A rough estimate of the area under the curve represented by \( f(x) = h(f(a) + f(a + h) + f(a + 2h) + f(a + 3h)) \). This formula generalizes to \( A \approx h \sum_{i=0}^{n-1} f(a + ih) \), where \( h = \frac{b-a}{n} \).

The problem with this approach is that \( h \) must be sufficiently small (or the number of panels, \( n \), must be sufficiently large) to avoid estimation errors. However, if \( h \) is too small the computation takes too long.

Write a program that prompts the user for the \( x \) bounds \( a \) and \( b \) and the desired number of panels, \( n \), for the computation. Have your program approximate the area using the formula above and display the result.

Have your solution use the following function whenever it needs a value for \( f(x) \).\(^2\)

```cpp
double f(double x) {
    static const double pi = atan(1.0)*4.0;
    return exp(-x*x/2)/sqrt(2*pi);
}
```

```java
import java.lang.Math;
public static double f(double x) {
    return Math.exp(-x*x/2)/Math.sqrt(2*Math.PI);
}
```

---

\(^1\) Image courtesy of http://www.physics.nau.edu/~hart/matlab/node53.html

\(^2\) This is the probability density function for the normal distribution with mean 0 and standard deviation 1.
5. Prime number predicate (20 Points)

Write a method that determines whether a given integer is prime. Also, write a program that inputs an integer from the user, calls the method you wrote, and then tells the user whether the input number is prime. A prime number is a positive integer that is only evenly divisible by 1 and itself. That is, the remainder of dividing a prime number by any positive number except 1 or itself is non-zero. You may assume the user enters a non-negative number. Note that 0 and 1 are not prime.

For example, if the user enters 293, the program should recognize that this number is prime and display “prime”. If the user enters 7429 the program should recognize that this number is not prime (it equals $17 \times 19 \times 23$) and display “not prime”.

6. What song is that? (20 Points)

You are writing an app that listens to music and identifies the song by finding a match in a database of known songs. While there are sophisticated approaches that identify the instruments and notes being played (essentially transcribing the music), a much simpler approach is sufficient in practice. It is only necessary to identify the pattern of particular features of the music, such as the beats, and match these features against a database of the known songs. As part of this feature identification, your job is to take a list of numbers indicating the times at which beats were detected and calculate some statistics about the time between the adjacent beats. In particular, you will calculate

- the range of the times between beats (call the minimum time between beats $r_0$, the maximum $r_4$, and the difference $d = r_4 - r_0$),
- the interquartile boundaries ($r_1 = r_0 + d \times 25\%$; $r_2 = r_0 + d \times 50\%$; $r_3 = r_0 + d \times 75\%$)
- and the number of durations in each of the 4 ranges, $c_1$ through $c_4$. For example, $c_3$ is the number of durations in the $3^{rd}$ interval such that $r_2 \leq x \leq r_3$. There is a small chance that some values will be exactly equal to $r_1...r_3$ depending on your implementation, these values might be counted twice; that is okay.

Write a program that accepts the beat times from a text file containing one time per line and outputs the information described above.

Consider the following example:

This is provided as a data file on the contest website with the filename “beat1.txt”.

Times: 0.00 0.24 0.75 1.02 1.25 1.73 2.46 2.72 2.85 3.11 3.25 3.49 4.52 5.01 5.51 5.75 5.87 6.13 6.39 6.66 6.79

Differences: 0.24 0.51 0.27 0.23 0.48 0.73 0.26 0.13 0.26 0.14 0.24 1.03 0.49 0.50 0.24 0.12 0.26 0.26 0.27 0.13

Sorted differences: 0.12 0.13 0.13 0.14 0.23 0.24 0.24 0.24 0.26 0.26 0.26 0.26 0.27 0.27 0.48 0.49 0.50 0.51 0.73 1.03

Initial calculations: $r_0 = 0.12$; $r_4 = 1.03$; $d = 0.91$; $r_1 = 0.3475$; $r_2 = 0.575$; $r_3 = 0.8025$

Number of values in each quartile, starting with the lowest: 14, 4, 1, 1

The sorted differences are shown for illustration; you are not required to produce a list of sorted differences.
7. Supply drops (40 Points)
You are given a file with 2-D coordinates of the centers of between 2 and 9 cities. For example...

A 0.0 0.0
B 5.0 3.4
C 8.4 -2.4
D 0.0 4.3
E 3.2 1.4

Each city has a 1-letter name.

An airplane must fly to all the cities and drop a package in a field at the center of each city. The plane may start at any city.

Calculate the shortest route (output order of cities and total distance).

Although certain simplifications can be made to the solution of this particular problem, problems similar to these are generally very time-consuming to solve since, to guarantee that you have the absolute shortest path for any set of coordinates, you must search all combinations of order. E.g., for 5 cities, there are 5! = 120 orders and for 9 cities, there are 9! = 362,880 orders. It is recommended that you solve this problem by searching all possibilities – the more efficient solutions are more difficult to implement.

The example above is provided as a data file on the contest website with the filename supply1.txt.
8. Image processing (40 Points)

Automated optical character recognition (OCR) software often needs to identify all of the contiguous elements in a monochrome image. This requires that the image be scanned pixel-by-pixel to determine when two “black” pixels are adjacent to each other. Pixels are considered adjacent when they touch horizontally, vertically, or diagonally. Each region is marked with a unique code so that it can be extracted later by the character recognition software. For example, the following image would be grouped as follows (NOTE: Your use of the specific classification symbols, A,B,C,D, may vary as long as each is unique):

| ** ** **** | AA AA BBBB |
| ** ** * | AA AA B |
| * * * *** | A A A BBB |
| * * * | A A B |
| ** ** **** | AA AA BBBB |
| *** ***** | CCC DDDDD |
| * * | C C D |
| * * *** | C C DDD |
| * * | C C D |
| *** ***** | CCC DDDDD |

Before Processing | After Processing

In this case the file was separated into four separate elements and each pixel of each element was labeled with a letter unique to that element. The actual letter is irrelevant as long as each is unique.

Write a program that prompts the user for a data file and extracts the number of rows and columns in the image from the first line. Each subsequent line will contain one row of the image with space used for a white pixel and “**” used for a black pixel. Find the separate elements in the image and display the processed image to the console. For your convenience, two sample files have been provided. `image1.txt` contains the example shown above and `image2.txt` contains a second image.
9. Degrees of separation (40 Points)

Facebook wants to add a feature that lets you enter someone’s email address and, if they are in the system, tells you how many “degrees of separation” they have from you. If you are friends with someone, you have 1 degree of separation. You have 0 degrees of separation from yourself. You have 2 degrees of separation from all your friends’ friends who are not your direct friends or yourself. In the figure at right, for example, “4” and “0” have 2 degrees of separation, “5” and “5” have 0 degrees of separation, and “2” and “4” have 3 degrees of separation. “6” and “3” have infinity degrees of separation.

Write a program that takes a file from the Facebook server that has a list of all “friend pairs” in the system. To protect user identities, the file assigns each user a unique number from 0 to N-1, where N is the number of registered users. The file contains N on the first line, and a friend pair separated by a space on each additional line. After reading the file and performing the needed calculations, your program will prompt the user to enter 2 user IDs and then report the degrees of separation between them. For example, for the given diagram, the file might contain

```
7
4 6
6 0
0 5
0 2
3 1
```

Hints

The above data file is provided on the contest website as separation1.txt

- Keep track of the distances between people by using a matrix or 2-D array.
- Since the distance from a to b is the same as from b to a, you only need to use half of the matrix (perhaps the half where a ≤ b), but you can use the entire matrix.

There are various ways to handle infinite degrees of separation, which is the case for every pair before considering any connections. For this program, use “9999” instead of infinity. When the result is “9999,” it will be understood that the users aren’t connected.

- Put “1”s in the matrix for each pair in the file. Now you have a matrix with 1s and 9999s but need to find where the 2s, 3s, etc. go.
- Iterate over the matrix as long as you find shorter connections. One correct algorithm is to consider each pair (a, b) and then consider all third parties c (you’ll probably need 3 or 4 loops to do this). If the degrees of separation (a, c) added to (b, c) are less than what you have for (a, b), then you’ve found a smaller number of degrees of separation for (a, b) and should update your matrix.