Programming Assignment 2

For each of the following problems, write a C++ program to implement the required functionalities, test it with several test cases and submit the source code using submit2000 on storm. These questions are taken from Problem Solving with C++, 10ed, Walter Savitch.

1. The Babylonian algorithm to compute the square root of a number **n** is as follows:

1) Make a guess at the answer (you can pick **n/2** as your initial guess), i.e, initialize **guess** to **n/2**
2) Compute **r=n/guess**
3) set **guess=(guess+r)/2**
4) Go back to step 2 for as many iterations as necessary. The more that step 2 and 3 are repeated, the closer guess will become to the square root of **n**.

Write a program that inputs a double for **n**, and iterates through the Babylonian algorithm 100 times. For a more challenged version, iterate until guess is within 1% of the previous guess, and outputs the answer as a double.

Requirement:

- You should define a function named squart_root that calculates and returns the square root of a number given as a double, and return the square root of the number.
- Include basic error checking in your code: for example, the input number cannot be negative.
- Name your program as sqrt.cpp, and submit it using: submit2000 LAB2 sqrt.cpp

****

2. Write a program that reads in ten whole numbers and that outputs the sum of all the numbers greater than zero, the sum of all the numbers less than zero, and the sum of all the numbers (whether positive, negative, or zero). The user enters the ten numbers just once each, and the user can enter them in any order (your program cannot ask the user to enter the positive numbers and negative numbers separately).

Requirement:

- Notice that you don't need to store the 10 numbers to implement the above functionalities. Imagine in situation where the input is 1 million whole numbers, it will be inefficient to store these numbers.
- Name your program as sum.cpp, and submit it using: submit2000 LAB2 sum.cpp

****

3. The flowchart (at the end of the document) contains a series of questions to determine the action in a soccer match. Write a program that asks the questions to the user, and outputs the recommended action for the user.

Requirement:

- Please use nested if-else statement to implement the decision tree. Pay attention to the indentation!
- You should allow the user to enter Y or y or yes or YES as input, N or n or no or no as input. (i.e., try to user friendly).
- Name your program as soccer.cpp. To submit: submit2000 LAB2 soccer.cpp

****

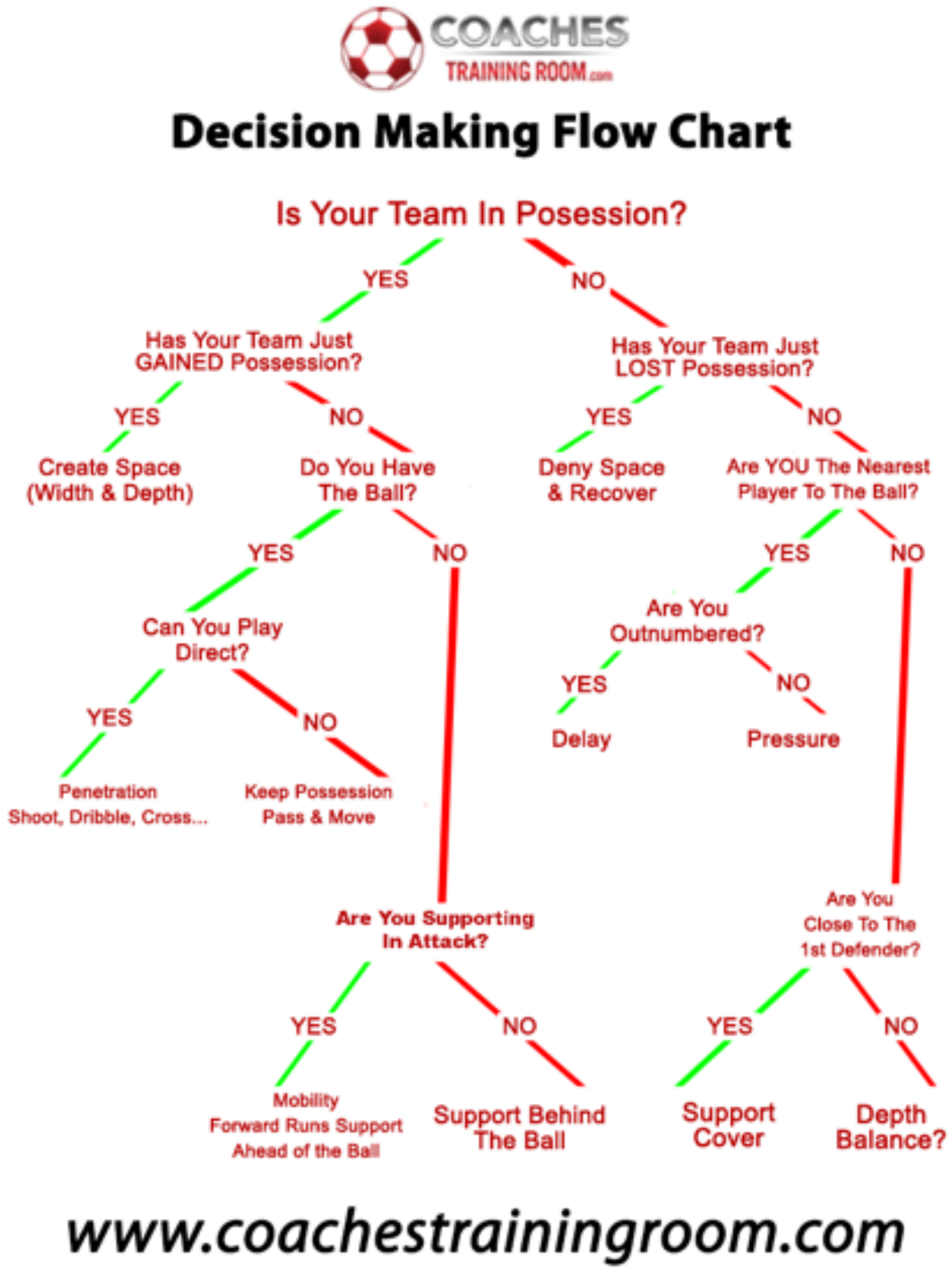
4. Write a program that finds and prints all prime numbers between 3 and 100. A prime number is a number such that 1 and itself are the only numbers that divides it (for example, 2, 3, 5, 6, …).

Hint:

- One way to solve this problem is to use a doubly nested loop. The outer loop can iterate from 2 to 100 while the inner loop checks to see if the counter value for the outer loop is prime.
- To check if a number is prime, you can loop a counter from 2 to the number, and see if the number can be evenly divided by the counter.

Requirement:

- Name your program as prime.cpp. To submit this problem: submit2000 LAB2 prime.cpp


COACHES
TRAINING ROOM.com
Decision Making Flow Chart
Is Your Team In Posession?
YES
NO
Has Your Team Just GAINED Possession?
Has Your Team Just LOST Possession?
YES
NO
YES
NO
Create Space (Width & Depth)
Do You Have The Ball?
Deny Space & Recover
Are YOU The Nearest Player To The Ball?
YES
NO
YES
NO
Can You Play Direct?
Are You Outnumbered?
YES
NO
YES
NO
Penetration Shoot, Dribble, Cross...
Keep Possession Pass & Move
Delay
Pressure
Are You Supporting In Attack?
Are You Close To The 1st Defender?
YES
NO
YES
NO
Mobility Forward Runs Support Ahead of the Ball
Support Behind The Ball
Support Cover
Depth Balance?
www.coachestrainingroom.com