Each of the following assignments is to be submitted to the autograder at gradescope.com. The assignment has a specific filename you must create or the grader will fail your work. You are allowed to submit as many times as you like to get full credit.

Lesson 3 (1 in Edx)

1. Hello World

Description
Write a program that prints Hello World

File Name
hello.cpp

Score
There is a single test worth 2 points

Lesson 4 (2 in Edx)

1. Coin Counter

Description
Write a program that asks the user to enter a number of quarters, dimes, nickels and pennies and then outputs the monetary value of the coins in the format of dollars and remaining cents.

Your program should interact with the user exactly as it shows in the following example:

Please enter the number of coins:
# of quarters: 20
# of dimes: 4
# of nickels: 13
# of pennies: 17
The total is 6 dollars and 22 cents

File Name
counter.cpp

Score

There are five tests each worth 2 points

2. Coin Converter

Description
Write a program that asks the user to enter an amount of money in the format of dollars and remaining cents. The program should calculate and print the minimum number of coins (quarters, dimes, nickels and pennies) that are equivalent to the given amount.

Hint: In order to find the minimum number of coins, first find the maximum number of quarters that fit in the given amount of money, then find the maximum number of dimes that fit in the remaining amount, and so on.

File Name
coins.cpp

Score
There are five tests each worth 2 points

For example, an execution should look like this:
Please enter the amount of money to convert:

# of dollars: 2
# of cents: 37
The coins are 9 quarters, 1 dimes, 0 nickels and 2 pennies

3. BMI Metric

Description

Body mass index (BMI) is a number calculated from a person's weight and height. According to the Centers for Disease Control and Prevention, the BMI is a fairly reliable indicator of body fatness for most people. BMI does not measure body fat directly, but research has shown that BMI correlates to direct measures of body fat, such as underwater weighing and dual-energy X-ray absorptiometry. The formula for BMI is

$$\text{Weight}/\text{Height}^2$$

Where weight is in kilograms and height is in meters.

Write a program that prompts for metric weight and height and outputs the BMI with two decimal places.

For example, an execution could look like this:

```
Please enter weight in kilograms: 50
Please enter height in meters: 1.58
BMI is: 20.03
```

File Name

bmimetric.cpp

Score

There are five tests each worth 2 points

4. BMI Imperial

Description

Body mass index (BMI) is a number calculated from a person's weight and height. According to the Centers for Disease Control and Prevention, the BMI is a fairly reliable indicator of body fatness for most people. BMI does not measure body fat directly, but research has shown that BMI correlates to direct measures of body fat, such as underwater weighing and dual-energy X-ray absorptiometry. The formula for BMI is

$$\text{Weight}/\text{Height}^2$$

Where weight is in kilograms and height is in meters.

Write a program that prompts for weight in pounds and height in inches, converts the values to metric, and then calculates the BMI with two decimal places. *Note: 1 pound is 0.453592 kilograms and 1 inch is 0.0254 meters.*

For example, an execution could look like this:

```
Please enter weight in pounds: 135
Please enter height in inches: 71
BMI is: 18.83
```

File Name

bmiimperial.cpp

Score

There are five tests each worth 2 points

Lesson 5 (3 in Edx)

1. Grade Classification

Description
Write a program that inputs two grades separated by a space.

If both grades are below a score of sixty then the program should output the message:
    Student Failed:(

If both grades are greater than or equal to ninety five then the program should output the message:
    Student Graduated with Honors:)

Otherwise the program should output the message:
    Student Graduated!

For example, an execution could look like this:

Please enter 2 grades, separated by a space:: 59 95
Student Graduated!

File Name
gradeclass.cpp

Score
There are five tests each worth 2 points

2. Counting Even and Odd Numbers

Description
Write a program that inputs four numbers separated by spaces. The program should count how many odd and even numbers there are. The program should then output one of three possibilities depending on how many even and odd numbers are entered:

1. more evens
2. more odds
3. same number of evens and odds

For example, an execution could look like this:

Please enter 4 positive integers, separated by a space: 2 3 5 7
mode odds

File Name
countoddeven.cpp

Score
There are five tests each worth 2 points

3. Leap Year Command Line Program

Description
Write a program for determining if a year is a leap year. In the Gregorian calendar system you can check if it is a leaper if it is divisible by 4 but not by 100 unless it is also divisible by 400.

For example, 1896, 1904, and 2000 were leap years but 1900 was not.
Write a program that takes in a year as input (as a command line argument) and returns the string "{year} was a leap year" if true and "{year} was not a leap year" if false.
Note: background on leap year https://en.wikipedia.org/wiki/Leap_year

Here is a positive example call to the program

```
.\isleapyearc 1896
```

output:
1896 was a leap year

Here is a negative example call to the program

```
.\isleapyearc 1897
```

output:
1897 was not a leap year

File Name
isleapyearc.cpp

Lesson 6 (4 in Edx)

1. Even Numbers

Description

Write a program that reads a positive integer n, and prints the first
n even numbers.

For example, one execution would look like this:
Please enter a positive integer: 3
2
4
6

File Name
evennumbers.cpp

Score
There are three tests each worth 2 points

2. Fibonacci

Description
Fibonacci number is a series of numbers in which each number is the sum of the two preceding numbers. The first two numbers in the series are the number 1. Write a program to print first n Fibonacci Numbers
For example, one execution would look like this:
Please enter a positive integer greater than 1: 4
1
1
2
3

File Name
fibonacci.cpp

Score
There are five tests each worth 2 points

Lesson 8(6 in Edx)

1. Leap Year Function

Description

Write a function for determining if a year is a leap year in the Gregorian calendar. The system is to check if it is divisible by 4 but not by 100 unless it is also divisible by 400.

For example, 1896, 1904, and 2000 were leap years but 1900 was not.
Write a function that takes in a year as input and returns True if the year is a leap year, return False otherwise.

Note: background on leap year https://en.wikipedia.org/wiki/Leap_year

The name of the function should be isleapyear and the function should take one parameter which is the year to test. Here is an example call to the function

```
int mybirthyear = 2000
If( isleapyear(mybirthyear)) {
   printf("Year %i  was a leap year",mybirthyear)
}
```

File Name
Isleapyearf.cpp

Function Signature
bool isleapyear(int inyear);

Score
There are five tests each worth 2 points

Note: You do not need any other code including the main method or any print statements. ONLY the *isleapyear* function is required. Otherwise, the autograder will fail and be unable to grade your code. (I.e., do not include the above example in your code.) The above example should be used to test your code but deleted or commented out upon submission.

2. BMI Metric Function

Description

Body mass index (BMI) is a number calculated from a person's weight and height. According to the Centers for Disease Control and Prevention, the BMI is a fairly reliable indicator of body fatness for most people. BMI does not measure body fat directly, but research has shown that BMI correlates to direct measures of body fat, such as underwater weighing and dual-energy X-ray absorptiometry. The formula for BMI is

$$\text{Weight/Height}^2$$

Where weight is in kilograms and height is in meters.

The name of the function should be named bmimetricf and the function should take two parameters which are the weight in kilograms and the height in meters to use to calculate the BMI.. Here is an example call to the function

```
int weight = 50;
float height = 1.58;
printf("BMI is: %3.2f",bmimetricf(weight, height)) ;
```

The output of the code above would be: BMI is: 20.03

File Name

bmimetricf.cpp

Function Signature

```
float bmimetricf(int weight, float height);
```

Score

There are five tests each worth 2 points

Note: You do not need any other code including the main method or any print statements. ONLY the *bmimetricf* function is required. Otherwise, the autograder will fail and be unable to grade your code. (I.e., do not include the above example in your code.) The above example should be used to test your code but deleted or commented out upon submission.

Lesson 9- Arrays (7 in edX)

1. Max In List Function

Description

Implement function max_val(lst), which returns the maximum value of the elements in the array.
For example, given an array lst: [-19, -3, 20, -1, 5, -25], the function
should return 20.

The name of the method should be maxinlst and the method should take two parameters. The first parameter is the array of values to test. The second parameter is the number of elements in the array. Here is an example call to the function

```
int lst[] = {-19, -3, 20, -1, 5, -25};
printf("%i",maxinlst(lst,6));
```

File Name
maxinlst.cpp

Function Signature

```
int maxinlst(int lst[], int size);
```

Score
There are three tests each worth 2 points

Note: You do not need to submit any other code including the main method or any print statements. ONLY the *maxinlst* function is required. Otherwise, the autograder will fail and be unable to grade your code. (I.e., do not include the above example in your code.) The above example should be used to test your code but deleted or commented out upon submission.

2. Max Absolute In List Function

Description

Implement function maxabsinlst(lst), which returns the maximum absolute value of the elements in the array.
For example, given a array lst: [-19, -3, 20, -1, 0, -25], the function should return 25.

The name of the method should be maxabsinlst and the method should take two parameters. The first parameter is the array of values to test. The second parameter is the number of elements in the array. Here is an example call to the function

```
int lst[] = {-19, -3, 20, -1, 5, -25};
printf("%i",maxinlst(lst,6));
```

Function Signature
int maxabsinlst(int lst[], int size);

File Name
maxabsinlst.cpp

Score
There are three tests each worth 2 points

Note: You do not need to submit any other code including the main method or any print statements. ONLY the *maxabsinlst* function is required. Otherwise, the autograder will fail and be unable to grade your code. (I.e., do not include the above example in your code.) The above example should be used to test your code but deleted or commented out upon submission.

3. Average of List Function

Description

Implement function avgoflst(lst), which returns the average value of the elements in the array.
For example, given a array lst: [19, 2, 20, 1, 0, 18], the function should return 10.

The name of the method should be avgoflst and the method should take two parameters. The first parameter is the array of values to test. The second parameter is the number of elements in the array. Here is an example call to the function

```
int lst[] = {19, 2, 20, 1, 0, 18};
printf("%i",avgoflst(lst,6));
```

Function Signature

```
float avgoflst(int lst[], int size);
```

File Name

avgoflst.cpp

Score

There are three tests each worth 2 points

Note: You do not need to submit any other code including the main method or any print statements. ONLY the avgof*lst* function is required. Otherwise, the autograder will fail and be unable to grade your code. (I.e., do not include the above example in your code.) The above example should be used to test your code but deleted or commented out upon submission.

Lesson 10 (8 in edX)

1. BMI Metric With Status Category

Description

Modify your earlier BMI Metric program to also display the CDC standard weight status categories:

The CDC standard weight status categories are:

BMI	Weight Status
Below 18.5	Underweight

18.5-24.9	Normal
25.0-29.9	Overweight
30.0 and above	Obese

For example, an execution could look like this:
Please enter weight in kilograms: 50
Please enter height in meters: 1.58
BMI is: 20.03, Status is Normal

File Name
bmimetric2.cpp

Score
There are five tests each worth 2 points

2. Cash Register

Description
Write a program that computes how much a customer has to pay after purchasing two items.

The price is calculated according to the following rules:
• Buy one get one half off promotion: the lower price item is half price.
• If the customer is a club card member, additional 10% off.
• Tax is added.

Inputs to the program include:
• Two items' prices
• Have club card or not (User enters 'Y' or 'y' for "yes"; 'N' or 'n' for "no")
• Tax rate (User enters the percentage as a number; for example, they enter 8.25 if the tax rate is 8.25%)

Program displays:
• Base price - the price before the discounts and taxes
• Price after discounts - the price after the buy one get one half off promotion and the member's discount, if applicable
• Total price – the amount of money the customer has to pay (after tax) printed with the precision of 2 decimal digits.

Hint: In order to print a number in a specific precision, you can use the round function passing 2 arguments to it. Use help(round) to get a brief explanation of

this function, and try playing with it, to better understand what it does.
To format to two decimal places you can use the string format function with the format of 2.2f.

For example, an execution could look like this:

Enter price of the first item: 10
Enter price of the second item: 20
Does customer have a club card? (Y/N): y
Enter tax rate, e.g. 5.5 for 5.5% tax: 8.25
Base price = 30.00
Price after discounts = 22.50
Total price = 24.36

File Name
cashregister.cpp

Score
There are five tests each worth 2 points

3. Call Cost Calculator

Description

Write a program that computes the cost of a long-distance call. The cost of the call is determined according to the following rate schedule:
• Any call started between 8:00 A.M. and 6:00 P.M., Monday through Friday, is billed at a rate of $0.40 per minute.
• Any call starting before 8:00 A.M. or after 6:00 P.M., Monday through Friday, is charged at a rate of $0.25 per minute.
• Any call started on a Saturday or Sunday is charged at a rate of $0.15 per minute.
The input will consist of the day of the week, the time the call started, and the length of the call in minutes.
The output will be the cost of the call.

Notes:
1. The time is to be input as 4 digit number, representing the time in 24-hour notation, so the time 1:30 P.M. is input as 1330
2. The day of the week will be read as one of the following three character string: ‘Mon’, ‘Tue’, ‘Wed’, ‘Thr’, ‘Fri’, ‘Sat’ or ‘Sun’
3. The number of minutes will be input as a positive integer.

For example, an execution could look like this:

Enter the day the call started at: Fri
Enter the time the call started at (hhmm): 2350
Enter the duration of the call (in minutes): 22
This call will cost $5.50

File Name
callcoster.cpp

Score
There are five tests each worth 2 points

4. String Splitter

Description

Read from the user a string containing
odd number of characters. Your program should:
a) Print the middle character.
b) Print the string up to but not including the middle character (i.e., the first half of the string).
c) Print the string from the middle character to the end (not including the middle character).

Sample output:
Enter an odd length string: Fortune favors the bold
Middle character: o
First half: Fortune fav
Second half: rs the bold

File Name
stringsplitter.cpp

Score
There are five tests each worth 2 points

5. Character Type

Description

Write a program that reads a character (string of length 1) from the user, and classifies it to one of the following: lower case letter, upper case letter, digit, or non-alphanumeric character

Sample output (4 different executions):
Enter a character: j
j is a lower case letter.
Enter a character: 7
7 is a digit.
Enter a character: ^
^ is a non-alphanumeric character.
Enter a character: C
C is an uppercase letter.

File Name
chartype.cpp

Score
There are five tests each worth 2 points

6. First Word Function

Description
Write a function that is given a phrase and returns the first word in that phrase.
For example, given ‘the quick brown fox’, your function should return ‘the’.

Here is an example call to the function
cout << firstword(“the quick brown fox”);

Function Signature
string firstword(string s);

File Name
firstword.cpp

Score
There are three tests each worth 2 points

Note: You do not need to submit any other code including the main method or any print statements. ONLY the *firstword* function is required (leave in your includes and namespace directives). Otherwise, the autograder will fail and be unable to grade your code. (I.e., do not include the above example in your code.) The above example should be used to test your code but deleted or commented out upon submission.

7. Remaining Words Function

Description
Write a function that is given a phrase and returns the phrase we get if we take out the first word from the input phrase.
For example, given ‘the quick brown fox’, your function should return ‘quick brown fox’

Here is an example call to the function
Cout << remainingwords(“the quick brown fox”);

Function Signature
string remainingwords(string s);

File Name
remainingwords.cpp

Score
There are three tests each worth 2 points

Note: You do not need to submit any other code including the main method or any print statements. ONLY the *remainingwords* function is required(leave in your includes and namespace directives). Otherwise, the autograder will fail and be unable to grade your code. (I.e., do not include the above example in your code.) The above example should be used to test your code but deleted or commented out upon submission.

C++ Data Structures Labs

Lesson 11 - Pointers & Dynamic Arrays

1. Add One Function

Description
In the C programming language there is no pass-by-reference syntax to pass a variable by reference to a function. Instead a variable is passed by pointer (just to be confusing, sometimes passing by pointer is referred to as pass-by-reference). This Practice Program asks you to do the same thing as C, which in practice would be simpler to implement using C++'s reference parameter syntax. Here is the header for a function that takes as
input a pointer to an integer:

Function Signature
void addOne(int *ptrNum)

Complete the function so it adds one to the integer referenced by ptrNum.
Write a main function where an integer variable is defined, give it an initial value, call addOne, and output the variable. It should be incremented by 1

File Name
addone.cpp

Score
There are three tests each worth 2 points

Note: You do not need to submit any other code including the main method or any print statements. ONLY the *addOne* function is required. Otherwise, the autograder will fail and be unable to grade your code. (I.e., do not include the above example in your code.) The above example should be used to test your code but deleted or commented out upon submission.

2. GetDoubles Function

Description
Write a function that takes a parameter that is an integer named numDoubles. Create a dynamic array that can store numDoubles doubles and make a loop that fills in each array entry with the value of the element number (starting with 1) divided by 3. Return the array from the function.

A sample execution with numDoubles equal to 4 would produce the array [0.3333, 0.6666, 1, 1.3333]

Function Signature
double* getDoubles(int numDoubles)

File Name
getdoubles.cpp

Score
There are three tests each worth 2 points

Note: You do not need to submit any other code including the main method or any print statements. ONLY the getDoubles function is required. Otherwise, the autograder will fail and be unable to grade your code. (I.e., do not include the above example in your code.) The above example should be used to test your code but deleted or commented out upon submission.

Lesson 12 - Recursion

1. GCD Recursive Function

Description
The greatest common divisor of integers x and y is the largest integer that evenly divides both x and y. Write a recursive function named gcd that returns the greatest common divisor of x and y. The gcd of x and y is defined recursively as follows: If y is equal to 0, then gcd(x,y) is x; otherwise gcd(x,y) is gcd(y,x%y) where % is remainder operator.

Here is an example call to the function
cout << gcd(15,5);

This example would print the number 5.

Function Signature
int gcd(int a, int b);

File Name
gcd.cpp

Score
There are three tests each worth 2 points

Note: You do not need to submit any other code including the main method or any print statements. ONLY the *gcd* function is required. Otherwise, the autograder will fail and be unable to grade your code. (I.e., do not include the above example in your code.) The above example should be used to test your code but deleted or commented out upon submission.

2. isPalindrome Recursive Function

Description
A palindrome is a word, number, phrase, or other sequence of characters which reads the same backward as forward, such as madam, racecar. Given an integer, write a function named is_palindrome that returns true if the given number is a palindrome, else false. For example, 12321 is palindrome, but 1451 is not palindrome.

Here is an example call to the function
cout << is_palindrome(12321);

This example would return true.

Function Signature
bool is_palindrome(int test);

File Name
ispalindrome.cpp

Score
There are three tests each worth 2 points

Note: You do not need to submit any other code including the main method or any print statements. ONLY the is_palindrome function is required. Otherwise, the autograder will fail and be unable to grade your code. (I.e., do not include the above example in your code.) The above example should be used to test your code but deleted or commented out upon submission.

Lesson 13 - Searching

1. Linear Search Function

Description
Given an integer and an array of integers, write a function named linear_search that returns the number of comparisons performed doing a linear search. The function should take 3 arguments:

- The number searched for
- The array of integers
- The number of elements in the array

If the number searched for is not in the array then the function should return the maximum number of searches to determine the element is not in the array.

Here is an example call to the function
int lst[] = {19, 2, 20, 1, 0, 18};
printf("%i",linear_search(20, lst, 6));

The above code would print 3.

Function Signature
int linear_search(int search_value, int lst[], int elements);

File Name
lsearch.cpp

Score
There are three tests each worth 2 points

Note: You do not need to submit any other code including the main method or any print statements. ONLY the linear_search function is required. Otherwise, the autograder will fail and be unable to grade your code. (I.e., do not include the above example in your code.) The above example should be used to test your code but deleted or commented out upon submission.

2. Binary Search Function

Description
Given an integer and a sorted array of integers, write a function named binary_search that returns the number of comparisons performed doing a binary search. The function should take 3 arguments:

- The number searched for
- The array of integers
- The number of elements in the array

If the number searched for is not in the array then the function should return the maximum number of searches to determine the element is not in the array.

Here is an example call to the function
int lst[] = {0, 1, 2, 18, 19, 20, 25};
printf("%i",binary_search(20, lst, 7));

The above code would print 2.

Function Signature
int binary_search(int search_value, int lst[], int elements);

File Name
bsearch.cpp

Score
There are three tests each worth 2 points

Note: You do not need to submit any other code including the main method or any print statements. ONLY the binary_search function is required. Otherwise, the autograder will fail and be unable to grade your code. (I.e., do not include the above example in your code.) The above example should be used to test your code but deleted or commented out upon submission.

Lesson 14 - Sorting

1. Swap Function

Description
The selection sort algorithm uses a function to swap the position of two elements in the array. Write the swap function for an integer selection sort

Here is an example call to the function
```
int lst[] = {19, 2, 20, 1, 0, 18};
swap(&lst[0],&lst[1]);
```

After the above code executes the array would look like this:.
{2, 19}

Function Signature
```
void swap(int *xp, int *yp) ;
```

File Name
swap.cpp

Score
There are three tests each worth 2 points

Note: You do not need to submit any other code including the main method or any print statements. ONLY the swap function is required. Otherwise, the autograder will fail and be unable to grade your code. (I.e., do not include the above example in your code.) The above example should be used to test your code but deleted or commented out upon submission.

2. Selection Sort Function

Description
Write the selection sort algorithm using the swap function developed in the previous problem. You should not upload the swap function but you must call the swap function to receive credit for the problem. You should include a header file named cpluspluslabs.h that has the signature of the swap function.

Here is an example call to the function
int lst[] = {19, 2, 20, 1, 0, 18};
selection_sort(lst,6);

After the above code executes the array would look like this:.
{0,1, 2, 18, 19, 20}

Function Signature
void selection_sort(int arr[], int elements);

File Name
selection.cpp

Score
There are three tests each worth 2 points

Note: You do not need to submit any other code including the main method or any print statements. ONLY the selection_sort function is required. Otherwise, the autograder will fail and be unable to grade your code. (I.e., do not include the above example in your code.) The above example should be used to test your code but deleted or commented out upon submission.

3. Merge Function

Description
The merge sort algorithm uses a function to merge two subarrays into one sorted array. In this lab you are writing the merge function. The function should take four parameters:

- Array
- Starting Index of first subarray
- Ending Index of first subarray
- Ending Index of second subarray

Here is an example call to the function
int lst[] = {0, 2, 1, 18, 20};
merge(lst,0,1,3);

After the above code executes the array would look like this:.
{0, 1, 2, 18, 20};

Function Signature
void merge(int arr[], int left, int middle, int right)

File Name
merge.cpp

Score
There are three tests each worth 2 points

Note: You do not need to submit any other code including the main method or any print statements. ONLY the merge function is required. Otherwise, the autograder will fail and be unable to grade your code. (I.e., do not include the above example in your code.) The above example should be used to test your code but deleted or commented out upon submission.

4. Merge Sort Function

Description
Write the merge sort algorithm using the merge function developed in the previous problem. You should not upload the merge function but you must call the merge function to receive credit for the problem. You should include a header file named cpluspluslabs.h that has the signature of the megere function. The merge sort function should take three parameters

1. Array to sort
2. Left Index to start of data
3. Right Index to end of data

Here is an example call to the function
int lst[] = {19, 2, 20, 1, 0, 18};
merge_sort(lst,0,5);

After the above code executes the array would look like this:.
{0,1, 2, 18, 19, 20}

Function Signature
void merge_sort(int arr[], int left, int right)

File Name
mergesort.cpp

Score
There are five tests each worth 2 points

Note: You do not need to submit any other code including the main method or any print statements. ONLY the merge_sort function is required. Otherwise, the autograder will fail and be unable to grade your code. (I.e., do not include the above example in your code.) The above example should be used to test your code but deleted or commented out upon submission.

Lesson 15 - Object Oriented Programming

1. ShowTicket Class

A theatre sells seats for shows and needs a system to keep track of the seats they have sold tickets for. Define a class for a type called ShowTicket.

The class should contain fields for the row, seat number, and whether the ticket has been sold or not. Define a constructor which accepts as arguments the row and seat number and sets the sold status to false in the constructor initialization section.

Include the following member functions

1. A function to check if the ticket has been sold with this signature:
   bool is_sold();

2. A function to update the ticket status to sold with this signature:

void sell_seat();

3. A function to print the row, seat number, and sold status delimited by a space with this signature:
   string print_ticket();

An example use of your class follows:

```
int main () {
  ShowTicket myticket1(“AA”,”101”);
  ShowTicket myticket2(“AA”,”102”);

  if(!myticket1.is_sold())
          myticket1.sell_seat ();
  cout <<  myticket1.print_ticket() << endl;
  cout <<  myticket2.print_ticket() << endl;
  return 0;
}
```

The output from this sample would be:
AA 101 sold
AA 102 available

File Name
showticket.h

Score
There are three tests each worth 2 points

Note: You do not need to submit any other code including the main method or any print statements. ONLY the ShowTicket class is required. Otherwise, the autograder will fail and be unable to grade your code. (I.e., do not include the above example in your code.) The above example should be used to test your code but deleted or commented out upon submission.

2. ShowTickets Class

A theatre sells seats for shows and needs a system to keep track of the seats they have sold tickets for. Define a class for a type called ShowTickets.

The class should contain a collection field for the rows, seat numbers, and whether the

tickets have been sold or not. Your class only needs to support 10 sold tickets. Assume a ticket you have not referenced before has a sold status of false.

Include the following member functions

4. A function to check if the ticket has been sold with this signature:
   bool is_sold(string row, string seat);

5. A function to update the ticket status to sold with this signature:
   void sell_seat(string row, string seat);

6. A function to print the row, seat number, and sold status delimited by a space with this signature:
   string print_ticket(string row, string seat);

An example use of your class follows:

```
int main () {
  ShowTickets myticket;

  if(!myticket.is_sold(“AA”,”101”))
          myticket.sell_seat (“AA”,”101”);
  cout <<  myticket.print_ticket(“AA”,”101”) << endl;
  cout <<  myticket.print_ticket(“AA”,”102”) << endl;
  return 0;
}
```

The output from this sample would be:

```
AA 101 sold
AA 102 available
```

File Name

showtickets.h

Score

There are three tests each worth 2 points

Note: You do not need to submit any other code including the main method or any print statements. ONLY the ShowTickets class is required. Otherwise, the autograder will fail and be unable to grade your code. (I.e., do not include the above example in your code.) The above example should be used to test your code but deleted or commented out upon submission.

3. SportTicket Class

A football club needs a system to keep track of the seats they have sold tickets for. Define a class for a type called SportTicket that inherits from the ShowTicket class you defined earlier. You should include the showticket.h file in your code but not upload the file.

The class should add a new boolean field to store if beer has been purchased at that seat. Define a constructor which accepts as arguments the row and seat number and sets the sold status and beer sold fields to false in the constructor initialization section.

The showticket class has the following protected members:

- string row;
- string seat;

Include the following new member functions

1. A function to check if the ticket has been sold with this signature:
   bool beer_sold();
2. A function to update the beer sold status to sold with this signature:
   void sell_beer();
3. override the print_ticket method to add if beer has been sold or not.

An example use of your class follows:

```
int main () {
  SportTicket myticket1(“AA”,”101”);
  SportTicket myticket2(“AA”,”102”);

  myticket1.sell_seat();
  myticket2.sell_seat();
  myticket2.sell_beer();
  cout <<  myticket1.print_ticket() << endl;
  cout <<  myticket2.print_ticket() << endl;
  return 0;
}
```

The output from this sample would be:

```
AA 101 sold nobeer
AA 102 sold beer
```

File Name
sportticket.h

Score
There are three tests each worth 2 points

Note: You do not need to submit any other code including the main method or any print statements. ONLY the SportTicket class is required (The grader already has the ShowTicket class). Otherwise, the autograder will fail and be unable to grade your code. (I.e., do not include the above example in your code.) The above example should be used to test your code but deleted or commented out upon submission.

4. TicketSales Class

You have been so successful in your previous object oriented programming you decide to develop a computer software system to compete with Ticketmaster

You then add a new class called TicketSales. This new class can print either a ShowTicket or a SportsTicket. You should include the header file sportsticket.h in your

Include the following new member functions

1. A function to print either a ShowTicket or a SportTicket with this signature:
   string print_ticket(ShowTicket *myticket);

An example use of your class follows:

```
int main () {
  TicketSales ts;
  ShowTicket myticket1(“AA”,”101”);
  SportTicket myticket2(“AA”,”102”);

  myticket1.sell_seat();
  myticket2.sell_seat();
  myticket2.beer_sold();
  cout <<  ts.print_ticket(&myticket1);
  cout <<  ts.print_ticket(&myticket2);

  return 0;
}
```

The output from this sample would be:
AA 101 sold
AA 102 sold beer

File Name
ticketsales.h

Score
There are three tests each worth 2 points

Note: You do not need to submit any other code including the main method or any print statements. ONLY the class TicketSales is required (The grader already has the ShowTicket and SportTicket classes). Otherwise, the autograder will fail and be unable to grade your code. (I.e., do not include the above example in your code.) The above example should be used to test your code but deleted or commented out upon submission.

Lesson 16 - File Processing

1. CSV Export Function

Description
A CSV is a comma-separated values file, which allows data to be saved in a tabular format. CSVs look like a garden-variety spreadsheet but with a . csv extension. CSV files can be used with most any spreadsheet program, such as Microsoft Excel or Google Spreadsheets. Each string has commas separating fields and a carriage return and linefeed at the end of each record. Write a csv_export function that takes in a two dimensional array of data and creates a file with the name specified in the parameter. The function will take a parameter (2nd for the number of columns). You can also assume 10 columns or less.

Here is an example call to the function

```
string data[10][10] =
{{aspen,olmsted,aspen@pleasedonotemail.com},{sally,jones,sally@gmail.com},{fred,smi
th,fsmith@aol.com}};
export_csv(data,3,”customers.csv”);
```

After the above code executes the file contents would look like this:.

```
aspen,olmsted,aspen@pleasedonotemail.com
sally,jones,sally@gmail.com
fred,smith,fsmith@aol.com
```

Function Signature

```
void csv_export(string data[][3], int records,int columns, string filename);
```

File Name

csvexport.cpp

Score

There are three tests each worth 2 points

Note: You do not need to submit any other code including the main method or any print statements. ONLY the csv_export function is required. Otherwise, the autograder will fail and be unable to grade your code. (I.e., do not include the above example in your code.) The above example should be used to test your code but deleted or commented out upon submission.

2. CSV Import Function

Description

A CSV is a comma-separated values file, which allows data to be saved in a tabular format. CSVs look like a garden-variety spreadsheet but with a . csv extension. CSV files can be used with most any spreadsheet program, such as Microsoft Excel or Google Spreadsheets. Each string has commas separating fields and a carriage return and linefeed at the end of each record. Write a csv_import function that reads a file and populates a two-dimensional array of data. For this exercise you can assume the array has space allocated for ten rows or more. The function should only load the first ten records. The function will take a parameter (2nd for the number of columns). You can also assume 10 columns or less.

Here is an example call to the function

```
string data[10][10];
```

```
Int records;
string data[10][10] = {{"aspen","olmsted","aspen@pleasedonotemail.com"}
      ,{"sally","jones","sally@gmail.com"}
      ,{"fred","smith","fsmith@aol.com"}}
csv_import(data,3,&records,"customers.csv");
```

After the above code executes the file contents would be in the array and the variable records would hold the number of rows that were in the file.

Function Signature
void csv_import(string data[][10], int columns, int *records, string filename);

File Name
csvimport.cpp

Score
There are three tests each worth 2 points

Note: You do not need to submit any other code including the main method or any print statements. ONLY the csv_import function is required. Otherwise, the autograder will fail and be unable to grade your code. (I.e., do not include the above example in your code.) The above example should be used to test your code but deleted or commented out upon submission.

Lesson 17 - Linked Lists

1. Add Node Tail Function

Description
Using the node definition below, write a function that takes parameters for the three data elements and adds the node to the end of the linked list. The definition of the Node is in a header file called cpluspluslabs.h that you should include but not submit it with your solution.

Use this definition for the nodes in the linked list:

```
class NODE {
public:
        string firstname;
        string lastname;
        string email;
```

```
        NODE* next;
};
```

Here is an example call to the function

```
NODE* data = nullptr;
add_node_tail(&data,”aspen”,”olmsted”,”aspeno@aol.com”);
```

After the above code executes the record will be the only entry in the linked list..

Function Signature
void add_node_tail(NODE** data, string firstname,string lastname, string email);

File Name
addnodet.cpp

Score
There are three test each worth 2 points

Note: You do not need to submit any other code including the main method or any print statements. ONLY the add_node_tail function is required. Otherwise, the autograder will fail and be unable to grade your code. (I.e., do not include the above example in your code.) The above example should be used to test your code but deleted or commented out upon submission.

2. Add Node Head Function

Description
Using the node definition below, write a function that takes parameters for the three data elements and adds the node to the beginning of the linked list. The definition of the Node is in a header file called cpluspluslabs.h that you should include but not submit it with your solution.

Use this definition for the nodes in the linked list:

```
class NODE {
public:
        string firstname;
        string lastname;
        string email;
        NODE* next;
};
```

Here is an example call to the function

```
NODE* data = nullptr;
add_node_head(&data,”aspen”,”olmsted”,”aspeno@aol.com”);
add_node_head(&data,”sally”,”jones”,”sjones@aol.com”);
```

After the above code executes there will be two records in the linked list with Sally’s recording coming first..

Function Signature
void add_node_head(NODE** data, string firstname,string lastname, string email);

File Name
addnodeh.cpp

Score
There are three tests each worth 2 points

Note: You do not need to submit any other code including the main method or any print statements. ONLY the add_node_head function is required. Otherwise, the autograder will fail and be unable to grade your code. (I.e., do not include the above example in your code.) The above example should be used to test your code but deleted or commented out upon submission.

3. CSV Import 2 Function

Description
Modify your earlier implementation of the CSV import function so instead of updating a two dimensional array it loads the data from the file into a linkedList.

Use this definition for the nodes in the linked list:

```
class NODE {
public:
        string first_name;
        string last_name;
        string email;
        NODE* next;
};
```

Here is an example call to the function

```
NODE* data = nullptr;
csv_import2(&data,"customers.csv");
```

After the above code executes the file contents would be in the linked list.

Function Signature
void csv_import2(NODE** data, string filename);

File Name
csvimport2.cpp

Score
There are five tests each worth 2 points

Note: The csv input file will have 3 columns to match the NODE class. This is different from the previous implementation that passed the number of columns as a parameter. You do not need to submit any other code including the main method or any print statements. ONLY the csv_import2 function is required. Otherwise, the autograder will fail and be unable to grade your code. (I.e., do not include the above example in your code.) The above example should be used to test your code but deleted or commented out upon submission.

Lesson 18 - Stacks and Queues

1. Push Method

Description
Using the starting code below, write the push method for a Stack class that stores integers in an array. If the caller tries to push more data than the MAX constant then the function should return false.

Note: The class fields are public only so the grader can check your code. In a real life implementation these should be private.

Use this code to implement your method:

```
#define MAX 1000
```

```
class Stack {

public:
    int top;
    int a[MAX]; // Maximum size of Stack

    Stack() { top = -1; }
    bool push(int x);
    int pop();
    int peek();
    bool isEmpty();
};
```

Here is an example call to the function

```
Stack mystack;
mystack.push(10);
```

After the above code executes the stack will contain the number 10 and the top field will contain the value 0.

Function Signature
bool push(int x);

File Name
stack.h

Score
There are three tests each worth 2 points

Note: You do not need to submit any other code including the main method or any print statements. ONLY the Stack class is required. Otherwise, the autograder will fail and be unable to grade your code. (I.e., do not include the above example in your code.) The above example should be used to test your code but deleted or commented out upon submission.

2. Peek Method

Description

Using the started code below, write the peek method for a Stack class that returns the top integer in the stack. If there is no data on the stack the peek method should return 0. You need to have a working push method in your stack so please make sure you have successfully completed the earlier assignments.

Note: The class fields are public only so the grader can check your code. In a real life implementation these should be private.

Use this code to implement your method:

```
#define MAX 1000

class Stack {

public:
   int top;
   int a[MAX]; // Maximum size of Stack

   Stack() { top = -1; }
   bool push(int x);
   int pop();
   int peek();
   bool isEmpty();
};
```

Here is an example call to the function

```
Stack mystack;
mystack.push(10);
printf("%i",mystack.peek());
```

After the above code executes the stack will contain the number 10 and the top field will contain the value 0. The number 10 will have been printed to the screen.

Function Signature

int peek();

File Name

stack.h

Score

There are three tests each worth 2 points

Note: You do not need to submit any other code including the main method or any print statements. ONLY the Stack class is required. Otherwise, the autograder will fail and be unable to grade your code. (I.e., do not include the above example in your code.) The above example should be used to test your code but deleted or commented out upon submission.

3. Pop Method

Description

Using the started code below, write the pop method for a Stack class that returns the top integer in the stack and removes it from the stack. If there is no data on the stack the pop method should return 0. You need to have a working push method in your stack so please make sure you have successfully completed the earlier assignments.

Note: The class fields are public only so the grader can check your code. In a real life implementation these should be private.

Use this code to implement your method:

```
#define MAX 1000

class Stack {

public:
   int top;
   int a[MAX]; // Maximum size of Stack

   Stack() { top = -1; }
   bool push(int x);
   int pop();
   int peek();
   bool isEmpty();
};
```

Here is an example call to the function

```
Stack mystack;
mystack.push(10);
printf("%i",mystack.pop());
```

After the above code executes the stack will be empty and the top field will contain the value -1. The number 10 will have been printed to the screen.

Function Signature
int pop();

File Name
stack.h

Score
There are three tests each worth 2 points

Note: You do not need to submit any other code including the main method or any print statements. ONLY the Stack class is required. Otherwise, the autograder will fail and be unable to grade your code. (I.e., do not include the above example in your code.) The above example should be used to test your code but deleted or commented out upon submission.

4. isEmpty Method

Description
Using the started code below, write the isEmpty method for a Stack class that returns true if the stack is empty or false if it is now. You need to have a working push and pop methods in your stack so please make sure you have successfully completed the earlier assignments.

Note: The class fields are public only so the grader can check your code. In a real life implementation these should be private.

Use this code to implement your method:

```
#define MAX 1000

class Stack {

public:
    int top;
    int a[MAX]; // Maximum size of Stack

    Stack() { top = -1; }
    bool push(int x);
    int pop();
    int peek();
    bool isEmpty();
};
```

Here is an example call to the function

```
Stack mystack;
mystack.push(10);
if(stack.isEmpty())
  printf(“No Data”);
else
  printf(“Data”);
```

The above code prints the string “Data" to the screen.

Function Signature
bool isEmpty();

File Name
stack.h

Score
There are three tests each worth 2 points

Note: You do not need to submit any other code including the main method or any print statements. ONLY the Stack class is required. Otherwise, the autograder will fail and be unable to grade your code. (I.e., do not include the above example in your code.) The above example should be used to test your code but deleted or commented out upon submission.

5. Enqueue Method

Description
Using the started code below, write the push method for a Queue class that stores integers in an array. If the caller tries to enqueue more data than the MAX constant then the method should return false.

Note: The class fields are public only so the grader can check your code. In a real life implementation these should be private.

Use this code to implement your method:

```
#define MAX 1000

class Queue {
```

```
public:
   int rear_value;
   int a[MAX]; // Maximum size of Queue

   Queue() { rear = -1; }
   bool enqueue(int x);
   int dequeue();
   int front();
   int rear();
};
```

Here is an example call to the function

```
Queue myqueue;
myqueue.enqueue(10);
```

After the above code executes the Queue will contain the number 10 and the rear field will contain the value 0.

Function Signature
bool enqueue(int x);

File Name
queue.h

Score
There are three tests each worth 2 points

Note: You do not need to submit any other code including the main method or any print statements. ONLY the Queue class is required. Otherwise, the autograder will fail and be unable to grade your code. (I.e., do not include the above example in your code.) The above example should be used to test your code but deleted or commented out upon submission.

6. Front Method

Description
Using the starting code below, write the front method for a Queue class that returns the integer at the front of the queue. If there is no data on the Queue the front method

should return 0. You need to have a working enqueue method in your Queue so please make sure you have successfully completed the earlier assignments.

Note: The class fields are public only so the grader can check your code. In a real life implementation these should be private.

Use this code to implement your method:

```
#define MAX 1000

class Queue {

public:
   int rear_value;
   int a[MAX]; // Maximum size of Queue

   Queue() { rear_value = -1; }
   bool enqueue(int x);
   int dequeue();
   int front();
   int rear();
};
```

Here is an example call to the function

```
Queue myqueue;
myqueue.enqueue(10);
printf("%i",myqueue.front());
```

After the above code executes the Queue will contain the number 10 and the front field will contain the value 0. The number 10 will have been printed to the screen.

Function Signature

Int front();

File Name

queue.h

Score

There are three tests each worth 2 points

Note: You do not need to submit any other code including the main method or any print statements. ONLY the Queue class is required. Otherwise, the autograder will fail and be unable to grade your code. (I.e., do not include the above example in your code.) The

above example should be used to test your code but deleted or commented out upon submission.

7. Rear Method

Description

Using the starting code below, write the rear method for a Queue class that returns the integer at the rear of the queue. If there is no data on the Queue the rear method should return 0. You need to have a working enqueue method in your Queue so please make sure you have successfully completed the earlier assignments.

Note: The class fields are public only so the grader can check your code. In a real life implementation these should be private.

Use this code to implement your method:

```
#define MAX 1000

class Queue {

public:
   int rear_value;
   int a[MAX]; // Maximum size of Queue

   Queue() { rear_value = -1; }
   bool enqueue(int x);
   int dequeue();
   int front();
   int rear();
};
```

Here is an example call to the function

```
Queue myqueue;
myqueue.enqueue(10);
printf("%i",myqueue.rear());
```

After the above code executes the Queue will contain the number 10 and the front field will contain the value 0. The number 10 will have been printed to the screen.

Function Signature

Int rear();

File Name
queue.h

Score
There are three tests each worth 2 points

Note: You do not need to submit any other code including the main method or any print statements. ONLY the Queue class is required. Otherwise, the autograder will fail and be unable to grade your code. (I.e., do not include the above example in your code.) The above example should be used to test your code but deleted or commented out upon submission.

8. Dequeue Method

Description
Using the starting code below, write the dequeue method for a Queue class that returns the front integer in the Queue and removes it from the Queue. If there is no data on the Queue the dequeue method should return 0. You need to have a working enqueue method in your Queue so please make sure you have successfully completed the earlier assignments.

Note: The class fields are public only so the grader can check your code. In a real life implementation these should be private.

Use this code to implement your method:

```
#define MAX 1000

class Queue {

public:
   int rear_value;
   int a[MAX]; // Maximum size of Queue

   Queue() { rear_value = -1; }
   bool enqueue(int x);
   int dequeue();
   int front();
   int rear();
};
```

Here is an example call to the function

```
Queue myqueue;
myqueue.enqueue(10);
printf("%i",myqueue.dequeue());
```

After the above code executes the Queue will be empty and the rear field will contain the value -1. The number 10 will have been printed to the screen.

Function Signature
Int dequeue();

File Name
queue.h

Score
There are three tests each worth 2 points

Note: You do not need to submit any other code including the main method or any print statements. ONLY the Queue class is required. Otherwise, the autograder will fail and be unable to grade your code. (I.e., do not include the above example in your code.) The above example should be used to test your code but deleted or commented out upon submission.

Lesson 19 - Trees

1. BST Constructors

Description
Using the class skeleton definition below, write the two constructors that create a new BST node that will be used for an integer Binary Search Tree (BST). The default constructor sets the data to 0.

Note: The class fields are public only so the grader can check your code. In a real life implementation these should be private.

Use this definition for the BST Class:

```
class BST
{
   public:
```

```
        int data;
        BST *left, *right;

         BST();
        BST(int);
        void Insert(int);
        int nth_node(int n);
};
```

Here is an example call to the function

BST mytree(10);

After the above code executes the tree will have an entry for the integer 10 with left and right set to null;.

Constructor Signatures
BST();
BST (int);

File Name
bst.h

Score
There are three tests each worth 2 points

Note: You do not need to submit any other code including the main method or any print statements. ONLY the BST class is required. Otherwise, the autograder will fail and be unable to grade your code. (I.e., do not include the above example in your code.) The above example should be used to test your code but deleted or commented out upon submission.

2. BST Insert

Description
Using the class skeleton definition below, write the insert method that inserts that create a new BST node that will be used for an integer Binary Search Tree (BST). The two constructors must be functioning in your class so please make sure you have the earlier assignments working or you will not get credit for this assignment.

Note: The class fields are public only so the grader can check your code. In a real life implementation these should be private.

Use this definition for the BST Class:

```
class BST
{
    public:
        int data;
        BST *left, *right;

        BST();
        BST(int);
        void Insert(int);
        int nth_node(int n);
};
```

Here is an example call to the function

```
BST node(10);
node.Insert(20);
```

After the above code executes the tree will have an entry for the integer 10 with left set to null and right pointing to a node with data of 20.

Function Signatures
BST* Insert(int);

File Name
bst.h

Score
There are three tests each worth 2 points

Note: You do not need to submit any other code including the main method or any print statements. ONLY the BST class is required. Otherwise, the autograder will fail and be unable to grade your code. (I.e., do not include the above example in your code.) The above example should be used to test your code but deleted or commented out upon submission.

3. BST Nth Node Function

Description
Using the class skeleton definition below, write the nth_node method that returns the nth node in order from the integer Binary Search Tree (BST). The number n is passed to the

function. The two constructors and the insert method must be functioning in your class so please make sure you have the earlier assignments working or you will not get credit for this assignment.

Note: The class fields are public only so the grader can check your code. In a real life implementation these should be private.

Use this definition for the BST Class:

```
class BST
{
        Public:
        int data;
        BST *left, *right;

        BST();
        BST(int);
        void Insert(int);
        int nth_node(int);
};
```

Here is an example call to the function

```
BST node(10);
node.Insert(20);
node.Insert(30);
node.Insert(1);
node.Insert(2);
cout << node.nth_node(3)
```

The above code will print 10.

Function Signatures

```
int nth_node(int);
```

File Name

bst.h

Score

There are three tests each worth 2 points

Note: You do not need to submit any other code including the main method or any print statements. ONLY the BST class is required. Otherwise, the autograder will fail and be unable to grade your code. (I.e., do not include the above example in your code.) The

above example should be used to test your code but deleted or commented out upon submission.

4. AVL - New Node Function

Description
Using the node definition below, write a function that creates a new node for an AVL tree that stores integers. Left, right should be set to null and the height should be set to 1.

Use this definition for the nodes in the AVL tree:

```
class Node
{
  public:
    int key;
    Node *left;
    Node *right;
    int height;
};
```

The Node is defined in a header file called cpluspluslabs.h that you should include in your code but not upload in your solution.

Here is an example call to the function

```
Node *root = new_node(10);
```

After the above code executes the tree will have the value 10 in it..

Function Signature

```
Node* new_node(int key);
```

File Name
avl.cpp

Score
There are three tests each worth 2 points

Note: You do not need to submit any other code including the main method or any print statements. ONLY the new_node function is required. Otherwise, the autograder will fail and be unable to grade your code. (I.e., do not include the above example in your code.) The above example should be used to test your code but deleted or commented out upon submission.

5. AVL - Unbalanced Insert Function

Description

Using the node definition below, write a function that inserts a new node into an AVL tree that stores integers. This version of the function should not balance the tree. It should increment the height properly. Since you are not balancing the tree the return node is the same that is passed in unless it is passed a null value for node and it will create a new node and return it.

Use this definition for the nodes in the AVL tree:

```
class Node
{
   public:
   int key;
   Node *left;
   Node *right;
   int height;
};
```

The Node is defined in a header file called cpluspluslabs.h that you should include in your code but not upload in your solution.

Here is an example call to the function

```
Node *root = insertnb(NULL, 10);
root = insertnb(root, 20);
root = insertnb(root, 30);
```

After the above code executes the tree will have the value 3 nodes all going to the right.

Function Signature

```
Node* insertnb(Node* node, int key);
```

File Name
avl.cpp

Score
There are three tests each worth 2 points

Note: You do not need to submit any other code including the main method or any print statements. ONLY the insertnb function is required. Otherwise, the autograder will fail and be unable to grade your code. (I.e., do not include the above example in your code.) The above example should be used to test your code but deleted or commented out upon submission.

6. AVL - Left Rotate Function

Description
Using the node definition below, write a function that left rotates an AVL tree that stores integers. The node to the right becomes the parent, the node passed in becomes the left. It should fix the height properly of the new and old parent nodes. The new parent should be returned from the rotation.

Use this definition for the nodes in the AVL tree:

```
class Node
{
   public:
   int key;
   Node *left;
   Node *right;
   int height;
};
```

The Node is defined in a header file called cpluspluslabs.h that you should include in your code but not upload in your solution.

Here is an example call to the function

```
Node *root = insertnb(NULL, 10);
root = insertnb(root, 20);
root = insertnb(root, 30);
root = left_rotate(root);
```

After the above code executes the tree will have 20 as the root node.

Function Signature
Node *left_rotate(Node *x)

File Name
avl.cpp

Score
There are three tests each worth 2 points

Note: You do not need to submit any other code including the main method or any print statements. ONLY the left_rotate function is required. Otherwise, the autograder will fail and be unable to grade your code. (I.e., do not include the above example in your code.) The above example should be used to test your code but deleted or commented out upon submission.

7. AVL - Right Rotate Function

Description
Using the node definition below, write a function that right rotates an AVL tree that stores integers. The node to the left becomes the parent, the node passed in becomes the right node. It should fix the height properly of the new and old parent nodes. The new parent should be returned from the rotation.

Use this definition for the nodes in the AVL tree:

```
class Node
{
   public:
   int key;
   Node *left;
   Node *right;
   int height;
};
```

The Node is defined in a header file called cpluspluslabs.h that you should include in your code but not upload in your solution.

Here is an example call to the function

```
Node *root = insertnb(NULL, 30);
root = insertnb(root, 20);
root = insertnb(root, 10);
root = right_rotate(root);
```

After the above code executes the tree will have 20 as the root node.

Function Signature
Node *right_rotate(Node *x)

File Name
avl.cpp

Score
There are three tests each worth 2 points

Note: You do not need to submit any other code including the main method or any print statements. ONLY the right_rotate function is required. Otherwise, the autograder will fail and be unable to grade your code. (I.e., do not include the above example in your code.) The above example should be used to test your code but deleted or commented out upon submission.

8. AVL - Get Balance Function

Description
Using the node definition below, write a function that returns the balance for a node of an AVL tree that stores integers. The balance is defined as the height of the left node minus the height of the right node. Zero should be returned if null is passed in.

Use this definition for the nodes in the AVL tree:

```
class Node
{
   public:
   int key;
   Node *left;
   Node *right;
   int height;
```

};

The Node is defined in a header file called cpluspluslabs.h that you should include in your code but not upload in your solution.

Here is an example call to the function

```
Node *root = insertnb(NULL, 30);
root = insertnb(root, 20);
root = insertnb(root, 10);
printf("%i",get_balance(root));
```

After the above code will print -3.

Function Signature
Int get_balance(Node *x)

File Name
avl.cpp

Score
There are three tests each worth 2 points

Note: You do not need to submit any other code including the main method or any print statements. ONLY the get_balance function is required. Otherwise, the autograder will fail and be unable to grade your code. (I.e., do not include the above example in your code.) The above example should be used to test your code but deleted or commented out upon submission.

9. AVL - Insert Function

Description
Using the node definition below, write a function that inserts a new node into an AVL tree that stores integers. This version of the function should balance the tree and return the new root node. If it is passed a null value for node it should create a new node and return it.

Use this definition for the nodes in the AVL tree:

```
class Node
{
   public:
   int key;
   Node *left;
   Node *right;
   int height;
};
```

The Node is defined in a header file called cpluspluslabs.h that you should include in your code but not upload in your solution.

Here is an example call to the function

```
Node *root = insert(NULL, 10);
root = insert(root, 20);
root = insert(root, 30);
root = insert(root, 40);
root = insert(root, 50);
root = insert(root, 25);
```

After the above code executes the constructed AVL Tree would be

```
          30
          / \
        20 40
        / \     \
     10 25    50
```

Function Signature

```
Node* insert(Node* node, int key);
```

File Name

avl.cpp

Score

There are three tests each worth 2 points

Note: You do not need to submit any other code including the main method or any print statements. ONLY the insert function and any support functions you write are required.

Otherwise, the autograder will fail and be unable to grade your code. (I.e., do not include the above example in your code.) The above example should be used to test your code but deleted or commented out upon submission.