Program 7
Arrays: building, sorting, printing
(100 points)

Overview

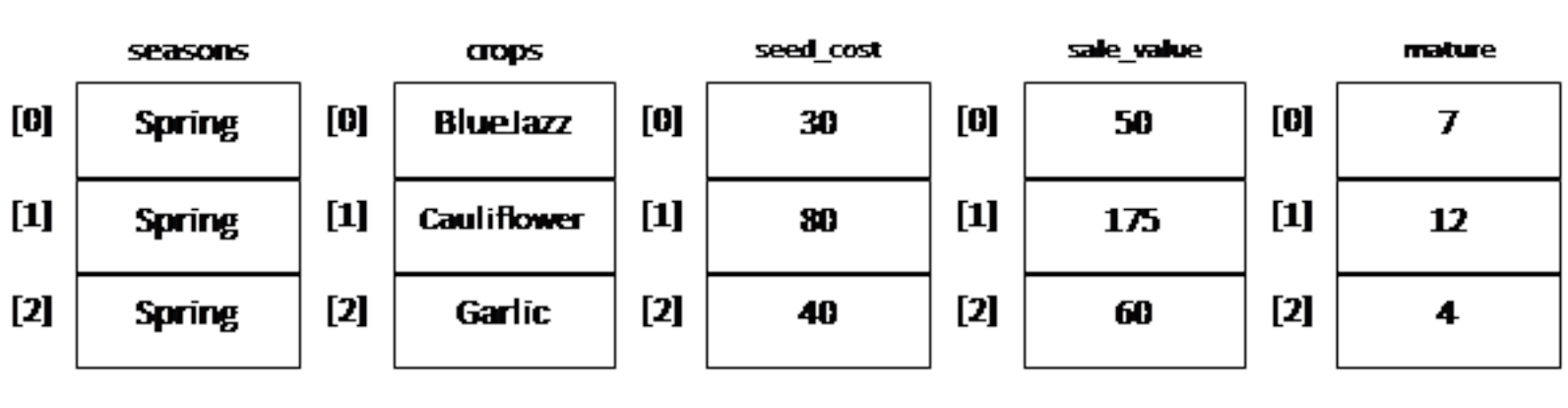
For this assignment, write a program that will process information regarding the profits to be earned from planting, nurturing, and selling crops such as is commonly done in farming simulation games such as Stardew Valley. Each crop will be described by the season in which they can be grown (Spring, Summer, and Fall), their name, the cost of their seed, the expected gold from the mature crops sale, and the number of days required for the crops to mature.

The information will be needed for later processing, so it will be stored in a set of arrays that will be displayed, sorted, and displayed (again).

For the assignment, declare five arrays, each of which will hold a maximum of 30 elements:

- one array of strings to hold the season name
- one array of strings to hold the crop name
- one array of integers to hold the cost of the seed
- one array of integers to hold the sale value of the mature crop
- one array of integers to hold the number of days it takes the crop to mature

The five arrays will be parallel arrays. This means that the information for one crop can be found in the same "spot" in each array. This also means that any time information is moved in one array the same move must be made to the other arrays in order to maintain data integrity.



Using an Input File in a CPP program

The data for this program will be read from an input file rather than using standard input (cin) or the random number generator. The file should be downloaded and saved. It is called crops.txt and can be found on Blackboard with the assignment write-up and here:

http://faculty.cs.niu.edu/~byrnes/csci240/pgms/crops.txt

The file consists of a set of records for the crops that are available for sale. Each record represents a single crop and contains: the season the crop is grown, the name of the crop, the cost of the seed, the expected sale value, the number of days for the crop to mature. A record in the file resembles the following:

```
Spring  BlueJazz        30      50      7
```

The entire file resembles the following:

```
Spring  BlueJazz        30      50      7
Spring  Cauliflower     80      175     12
Spring  Garlic  40      60      4
Spring  Kale    70      110     6
Spring  Parsnip 20      35      4
```

Note: It is okay to assume that if there is a season in a record, there will be a crop name, cost, sale value, and number of days to maturity.

To start working with an input file, a couple of extra include statements are needed for ***fstream*** and ***cstdlib***.

When a user entered the values for input, the C++ code used cin and the input operator (>>) to get the information from the keyboard and save it in a variable. cin is an input stream. To get information from a file, an input file stream is needed.

```
ifstream infile;        //input file stream variable this will be used instead of cin
```

The input file stream variable (infile in the example) will be used in place of cin when information is needed.

Now that the input file stream variable has been created, it needs to be "connected" with the text file that holds the information to be read. The "connection" is created by opening the file:

```
infile.open( "crops.txt" );      //open the file for reading
```

The open statement will try to open a file named crops.txt.

Windows Users: the crops.txt file MUST be located in the same directory as the CPP file.

Mac Users: there are three options available to handle the input file.

- **Option 1:** put in the set of double quotes ("") between the parenthesis for the open command, find where the file has been saved on the Mac, and then drag and drop the file in between the quotes. It should place the name of the file, including the complete path of where it is located on the Mac, between the quotes. ***If this option is used, before handing in the CPP file, remove the path name from the file name so that it only has crops.txt between the double quotes***.
- **Option 2 for newer versions of XCode (12+):** follow the directions in the PDF file found here http://faculty.cs.niu.edu/~byrnes/csci240/FilesAndXCode13.pdf
- **Option 3 for older versions of XCode:** follow the directions in the PDF file found here http://faculty.cs.niu.edu/~byrnes/csci240/FilesAndXCode.pdf

Once the file has been opened, it's important to make sure that it opened correctly. This is an important step because a file cannot be processed if it doesn't exist or open correctly. To test if the file opened correctly:

```
if( infile.fail() )       //if the input file failed to open
  {
  cout << endl << "crops.txt input file did not open" << endl;
  exit(-1)                //stop execution of the program immediately
  }
```

In the previous programs, the values have been read by executing a cout/cin combination. Since this program is reading input from a file, substitute the name of the input file stream (infile in the example) in place of cin. The cout statement is not needed with the data being read from a file.

So, if a program had something like:

```
cout << "Season? ";
cin >> season;
```

It should now be:

```
infile >> season;
```

When writing a read loop using data from a file, the loop should test to see if there is data in the file. One way to do this is to use the input file stream variable as a boolean variable:

```
while( infile )
```

As long as there is information in the file, the input file stream variable will remain valid (or true). Once the end of the data has been reached, the stream will become invalid (or false). ***Note: this test is ONLY successful AFTER an attempt has been made to read data from the file. This means that a standard read loop pattern that uses a priming and secondary read should be followed.***

Finally, once all the data has been read from the file, the file should be closed. To do this, execute the following:

```
infile.close();
```

onlinegdb Directions

For those that are using the onlinegdb C++ compiler (or any online compiler), this assignment will be a little more challenging because the online compilers cannot access the input file (crops.txt) that is being used.

Write the program so that it follows all the directions that are listed above.

The workaround so that this version of the assignment can be written and tested is to rely on cin to read the input and to copy/paste the values from the crops.txt file in the input area for the online compiler.

- Click on the input tab at the bottom of the compiler.
- Click the radio button with the **Text** label.
- Copy and paste the values from the crops.txt file into the input box that opened after the Text radio button was clicked.

Opening the file will fail because the crops.txt file cannot be accessed. This is okay. To make the program continue to run, comment out the exit command from above. DO NOT delete the command. Simply comment it out so that it can be made active once the program is ready to be handed in for grading.

```
if( infile.fail() )       //if the input file failed to open
  {
  cout << "crops.txt input file did not open" << endl;
//  exit(-1)                //stop execution of the program immediately
  }
```

Comment out the lines of code that read a value from the file and add one that reads from cin instead. For example:

```
//infile >> season;
cin >> season;                //this line will eventually be deleted
```

This will allow the program to be tested and keep the line of code that is needed for grading so it can be made active later.

Comment out the beginning of the while loop that tests to see if there is data in the file. Add the beginning of a while loop that tests cin.

```
//while( infile )
while( cin )                  //this line will eventually be deleted
```

The program can be run at this point.

Once the program is successful at reading information and putting it into the arrays, uncomment the exit command, the statements that use infile for reading, and the beginning of the while loop that tests to see if there is data in the file. Delete the commands that use cin (the ones with the "this line will eventually be deleted" message).

Basic Program Logic

Declare the five arrays and an integer variable to hold the number of crops. If any other variables are needed, declare them as well.

Fill the five arrays of by calling the **build()** function. The build() function returns the number of crops that it put into the arrays. This returned value should be saved in the integer variable that holds the number of crops.

Display the five arrays by calling the **print()** function. Make sure to pass the five arrays and the number of crops (the integer value returned from the build() call) to the print() function.

Sort the five arrays by calling the **sort()** function.

Display the sorted arrays by calling the **print()** function a second time.

Functions to write and use

Write and use the following functions in the program. More functions may be added if needed.

int build(string season[], string name[], int seed_cost[], int sell_value[], int harvest_time[])

This function will fill the five arrays with the values predefined within the crops.txt input file described above.

It returns an integer that is the number of crops read from the input file. **Note about the return value:** this value is important because it will be smaller than the maximum capacity of the arrays passed.

The function should start by declaring any variables that are needed. At a minimum, there should be a string variable to capture incoming string values, an integer to hold the incoming integer values, a second integer variable used to index each of the five arrays, and an input file stream.

Next, the function should attempt to open the input file and verify that it opened correctly.

Now that the file has been opened correctly, it's time to read the information, one value at a time, starting with the first season in the file.

In a loop that executes as long as there is information in the file, put the value that was read into the array that holds the seasons, read the remaining 4 values from the input file and put them into the corresponding arrays, update the index, and read the next season from the input file.

Finally, once all the data has been read from the file, close the file and return the number of crops that were placed in the arrays (Hint: subscript or index!).

void print(string season[], string name[], int seed_cost[], int sell_value[], int harvest_time[], int num_of_crops)

This function will display the information for all of the crops.

It takes as its arguments the array of strings that holds the seasons, the array of strings that holds the crop names, the array of integers that holds the cost of this crop's seed, the array of integers that holds the resale value of the matured crop, the array of integers that holds the number of days required for the crop to mature, and an integer that holds the number of crops that were defined from the input file.

It returns nothing.

The function should first display a report header.

In a loop that executes `num_of_crops` number of times, display all crop details in one line of output along with the gold profit per day.

The gold profit per day is calculated as follows:

```
Gold Profit Per Day = (sell_value - seed_cost) / harvest_time)
```

In the output provided below, each column is 16 characters wide, with 96 dashes used on the line after the column header. This number is deliberately mentioned to motivate using a certain method of creating this output, rather than cut 'n pasting the entire line into your source code. There are 30 spaces printed before the "Gold Profit Per..." title as well.

void sort(string season[], string name[], int seed_cost[], int sell_value[], int harvest_time[], int num_of_crops)

This function will sort the arrays in **DESCENDING** order based on the gold profit per day of each crop. Use the selection sort algorithm presented in lecture.

The arguments list here is exactly the same as with `void print` described above.

It returns nothing.

Remember that the five arrays are parallel arrays, meaning that elements in each array that have the same subscript correspond. Therefore, every time the algorithm swaps two elements in the crop name array, it must also swap the corresponding elements in the season, seed value, resale value and harvest time arrays.

Program Requirements:

1. The arrays should be able to hold 30 elements. Use a symbolic constant to represent the maximum size of the arrays.
2. It would be wise to store the column width for the output as a symbolic constant as well (16) and use that instead of "hard-coding" 16 each time. This way, we could easily re-size the width of the columns later with only one edit.
3. The arrays have the capability to hold 30 elements, however, that does not mean that it will all be used. This is the reason that the number of elements in the array is being passed to the print and sort functions. This value is the return value from the build function.
4. Add #include <fstream> and #include <cstdlib> at the top of the program.
5. Copy the input file and write the program so that it reads the data from the current directory (ie. don't specify a file path).
6. As with program 6, each of the functions MUST have a documentation box that describes the function. ***This is the last reminder about function documentation that will appear in the assignment write-ups.***
7. Hand in a copy of the source code (CPP file) using Blackboard.

Output

Here is the output for this assignment using the crops.txt file from above.

```
                              Gold Profit Per Day Report for All Crops

          Season       Crop Name       Seed Cost      Sell Value    Harvest Time Gold Profit/Day
------------------------------------------------------------------------------------------------
          Spring        BlueJazz              30              50               7            2.86
          Spring     Cauliflower              80             175              12            7.92
          Spring          Garlic              40              60               4            5.00
          Spring            Kale              70             110               6            6.67
          Spring         Parsnip              20              35               4            3.75
          Spring          Potato              50              80               6            5.00
          Spring         Rhubarb             100             220              13            9.23
          Spring           Tulip              20              30               6            1.67
          Spring    UnmilledRice              40              30               6           -1.67
          Summer           Melon              80             250              12           14.17
          Summer           Poppy             100             140               7            5.71
          Summer          Radish              40              90               6            8.33
          Summer      RedCabbage             100             260               9           17.78
          Summer       Starfruit             400             750              13           26.92
          Summer   SummerSpangle              50              90               8            5.00
          Summer       Sunflower             200              80               8          -15.00
          Summer           Wheat              10              25               4            3.75
            Fall        Amaranth              70             150               7           11.43
            Fall       Artichoke              30             160               8           16.25
            Fall            Beet              20             100               6           13.33
            Fall         BokChoy              50              80               4            7.50
            Fall       FairyRose             200             290              12            7.50
            Fall         Pumpkin             100             320              13           16.92
            Fall             Yam              60             160              10           10.00


                              Gold Profit Per Day Report for All Crops

          Season       Crop Name       Seed Cost      Sell Value    Harvest Time Gold Profit/Day
------------------------------------------------------------------------------------------------
          Summer       Starfruit             400             750              13           26.92
          Summer      RedCabbage             100             260               9           17.78
            Fall         Pumpkin             100             320              13           16.92
            Fall       Artichoke              30             160               8           16.25
          Summer           Melon              80             250              12           14.17
            Fall            Beet              20             100               6           13.33
            Fall        Amaranth              70             150               7           11.43
            Fall             Yam              60             160              10           10.00
          Spring         Rhubarb             100             220              13            9.23
          Summer          Radish              40              90               6            8.33
          Spring     Cauliflower              80             175              12            7.92
            Fall         BokChoy              50              80               4            7.50
            Fall       FairyRose             200             290              12            7.50
          Spring            Kale              70             110               6            6.67
          Summer           Poppy             100             140               7            5.71
          Spring          Potato              50              80               6            5.00
          Summer   SummerSpangle              50              90               8            5.00
          Spring          Garlic              40              60               4            5.00
          Spring         Parsnip              20              35               4            3.75
          Summer           Wheat              10              25               4            3.75
          Spring        BlueJazz              30              50               7            2.86
          Spring           Tulip              20              30               6            1.67
          Spring    UnmilledRice              40              30               6           -1.67
          Summer       Sunflower             200              80               8          -15.00
```

Extra Credit

For 5 points of extra credit, add functionality to this program that will prompt the user to input a season and an amount of gold investment, followed by determining how many of which crops will yield the most profit, and to quantify that profit amount. Additionally, we will need to check if whether there are enough days remaining in this season for the crops to reach maturity, otherwise they will be ruined by the season change. Further still, we need to make sure there's enough tilled land on the farm that these seeds can be sown into as well.

Please note that you can only get these extra credit points if you've first correctly completed the "main" objective of the assignment above.

Extra Credit Output Run 1

```
// Here would be the correct output from the "main" objective of the assignment

Seasons

1) Spring
2) Summer
3) Fall

Enter your choice:  (1 - 3): 1
How much gold are you investing? (0 - 100000): 14531
How many days are left in this season for crops to mature? (1 - 27): 8
How much tilled farmland is available to use? (0 - 128): 64

Purchase 64 Kale seeds, at the cost of 4480 gold, which will yield 2560.00 gold in profit after 6 days.
```

Explanation

The first Spring crop that is available is Rhubarb. However, it isn't a possibility because it takes too long to harvest (13 days when only 8 days are left).

The next Spring crop is Cauliflower. It also has a harvest time that is too long.

Kale is next. It's harvest time of 6 days allows enough time. A Kale seed costs 70 and with an investment of 14531, it is possible to purchase seeds. The maximum number of seeds that can be purchased with the current investment is 207. However, there is only room available for 64.

64 Kale seeds are purchased at a cost of 4480, leaving 10051 left to invest and no more farmland available. Since all the farmland is used, no more crops can be planted.

Extra Credit Output Run 2

```
// Here would be the correct output from the "main" objective of the assignment

Seasons

1) Spring
2) Summer
3) Fall

Enter your choice:  (1 - 3): 2
How much gold are you investing? (0 - 100000): 51006
How many days are left in this season for crops to mature? (1 - 27): 10
How much tilled farmland is available to use? (0 - 128): 99

Purchase 99 RedCabbage seeds, at the cost of 9900 gold, which will yield 15840.00 gold in profit after 9 days.
```

Explanation

The first Summer crop that is available is Starfruit. The harvest time is too long.

The next Summer crop is RedCabbage. There is enough time to harvest. At a cost of 100 and with an investment of 51006, it is possible to purchase 510 seeds. There is only room for 99 seeds.

99 RedCabbage seeds are purchased at a cost of 9900, leaving 41106 left to invest and no more farmland available. Since all the farmland is used, no more crops can be planted.

Extra Credit Output Run 3

```
// Here would be the correct output from the "main" objective of the assignment

Seasons

1) Spring
2) Summer
3) Fall

Enter your choice:  (1 - 3): 3
How much gold are you investing? (0 - 100000): 3499
How many days are left in this season for crops to mature? (1 - 27): 18
How much tilled farmland is available to use? (0 - 128): 128

Purchase 34 Pumpkin seeds, at the cost of 3400 gold, which will yield 7480.00 gold in profit after 13 days.
Purchase 3 Artichoke seeds, at the cost of 90 gold, which will yield 390.00 gold in profit after 8 days.
```

Explanation

The first Fall crop that is available is Pumpkin and there is enough time to harvest. At a cost of 100 and with an investment of 3499, it is possible to purchase 34 seeds. There is room for all 34 seeds.

34 Pumpkin seeds are purchased at a cost of 3400, leaving 99 left to invest and 94 pieces of farmland.

The next Fall crop is Artichoke. There is enough time to harvest. At a cost of 30 and with a current investment of 99, it is possible to purchase 3 seeds. There is room for all 3 seeds.

3 Artichoke seeds are purchased at a cost of 90, leaving 9 left to invest and 91 pieces of farmland.

The remaining Fall crops have costs higher than what is left to currently invest (9) so no more crops will be purchased.