Tatyana Vlaskin
Final Project

For the final project I took few programs from the book, and connected them to a farm theme. I've decided that was the easiest way to meet all or most of the requirements of the project. I've done the following problems from the book with few modifications.

CHAPTER 5: ARRAYS

15. Traditional password entry schemes are susceptible to "shoulder surfing" in which an attacker watches an unsuspecting user enter their password or PIN number and uses it later to gain access to the account. One way to combat this problem is with a randomized challenge-response system. In these systems, the user enters different information every time based on a secret in response to a randomly generated challenge. Consider the following scheme in which the password consists of a five-digit PIN number (00000 to 99999). Each digit is assigned a random number that is 1, 2, or 3. The user enters the random numbers that correspond to their PIN instead of their actual PIN numbers.

For example, consider an actual PIN number of 12345. To authenticate, the user would be presented with a screen such as

```
PIN: 0 1 2 3 4 5 6 7 8 9
NUM: 3 2 3 1 1 3 2 2 1 3
```

The user would enter 23113 instead of 12345. This does not divulge the password even if an attacker intercepts the entry because 23113 could correspond to other PIN numbers, such as 69440 or 70439. The next time the user logs in, a different sequence of random numbers would be generated, such as

```
PIN: 0 1 2 3 4 5 6 7 8 9
NUM: 1 1 2 3 1 2 2 3 3 3
```

Your program should simulate the authentication process. Store an actual PIN number in your program. The program should use an array to assign random numbers to the digits from 0 to 9. Output the random digits to the screen, input the response from the user, and output whether or not the user's response correctly matches the PIN number.

CHAPTER 6: STRUCTURES AND CLASSES

12. Your Community Supported Agriculture (CSA) farm delivers a box of fresh fruits and vegetables to your house once a week. For this Programming Project, define the class BoxOfProduce that contains exactly three bundles of fruits or vegetables. You can represent the fruits or vegetables as an array of type string . Add accessor and mutator functions to get or set the fruits or vegetables stored in the array. Also write an output function that displays the complete contents of the box on the console. Next, write a main function that creates a BoxOfProduce with three items randomly selected from this list:

Broccoli
Tomato
Kiwi
Kale
Tomatillo

This list should be stored in a text file that is read in by your program. For now you can assume that the list contains exactly five types of fruits or vegetables.

Do not worry if your program randomly selects duplicate produce for the three items. Next, the main function should display the contents of the box and allow the user to substitute any one of the five possible fruits or vegetables for any of the fruits or vegetables selected for the box. After the user is done with substitutions output the final contents of the box to be delivered.

CHAPTER 5: ARRAYS

THIS PROGRAM WILL BE SLIGHLY MODIFIED

14. The user will be asked to rate baskets that were delivered to them before. I am talking about baskets from the program **CHAPTER 6: STRUCTURES AND CLASSES** above. First the user will be asked how many baskets they want to grade. This will let me use dynamic array. Once the entry is accepted, the used will be asked to enter rating for each product in the basket. The ratings range from 1 (terrible) to 5 (excellent) and 0 – product was not in the basket. Once the user is done reviewing the products, they will be able to see their review in a table:

	Tomatillo	Broccoli	Tomato	Kiwi	Kale
1	1	5	2	2	3
2	1	1	1	1	1

At the same time, the user will be able to see the average rating for each product.

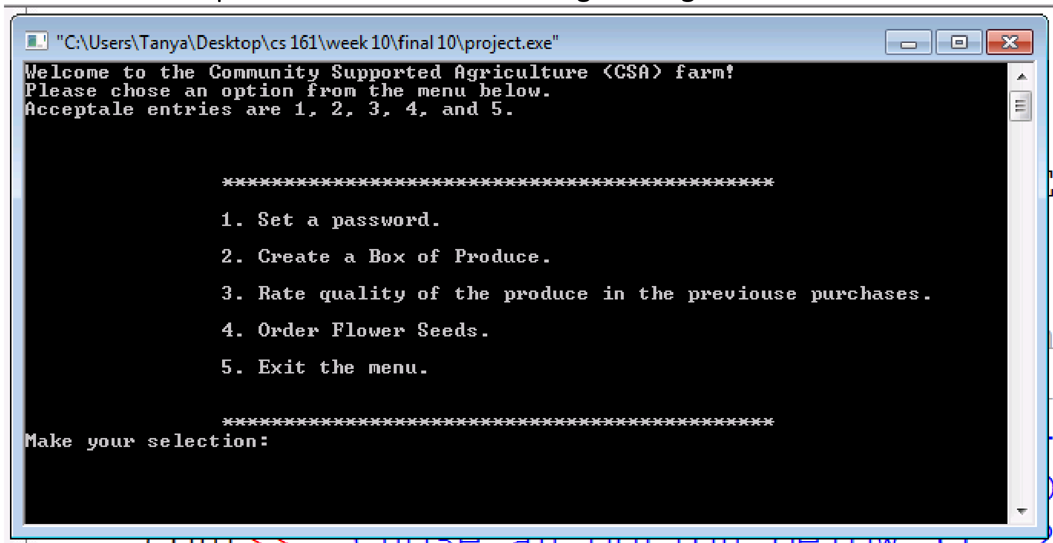
FINALLY:

Ill have another program that will present list of flower seed along with the prices to the user. The user will be asked what they want to buy. The user will be able to add one item at a time and once the user does not want to add any more items, a check will be presented to the user.

IMPLEMENTATION AND DESIGN:

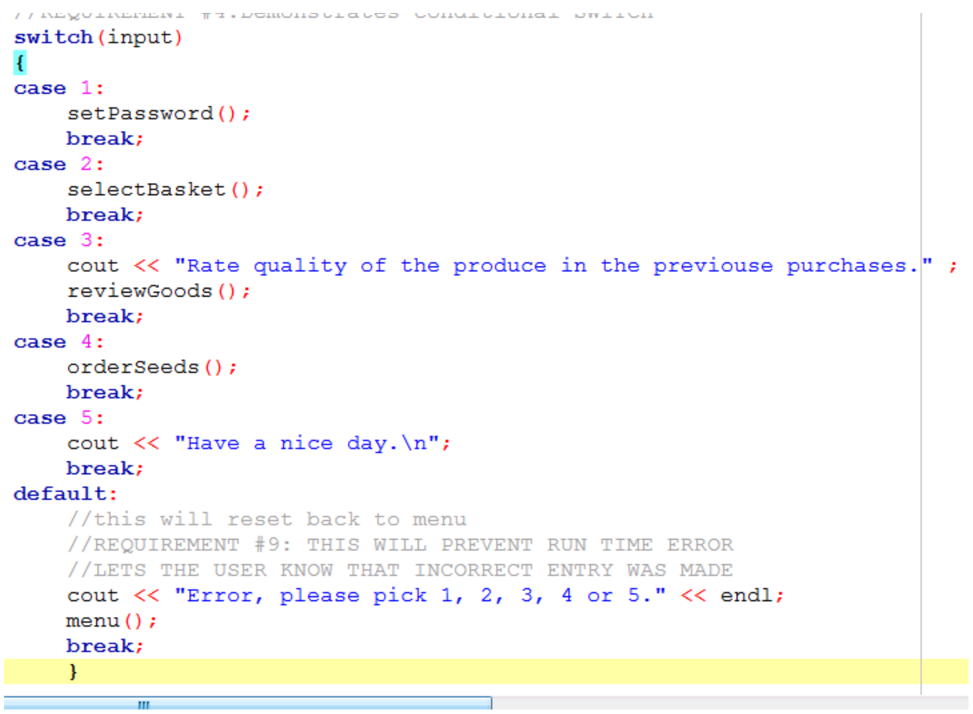
Per discussion board and per the requirements of the project, this was done simultaneously.

The user will be presented with the following message and menu:



```
Welcome to the Community Supported Agriculture (CSA) farm!
Please chose an option from the menu below.
Acceptale entries are 1, 2, 3, 4, and 5.


                **********************************************

                1. Set a password.

                2. Create a Box of Produce.

                3. Rate quality of the produce in the previouse purchases.

                4. Order Flower Seeds.

                5. Exit the menu.

                **********************************************
Make your selection:
```

The input of the user will be verified and if the user made invalid input, the user will be asked to make a different entry. One the entry is accepted, the following switch statement will be entered, to implement the corresponding function:

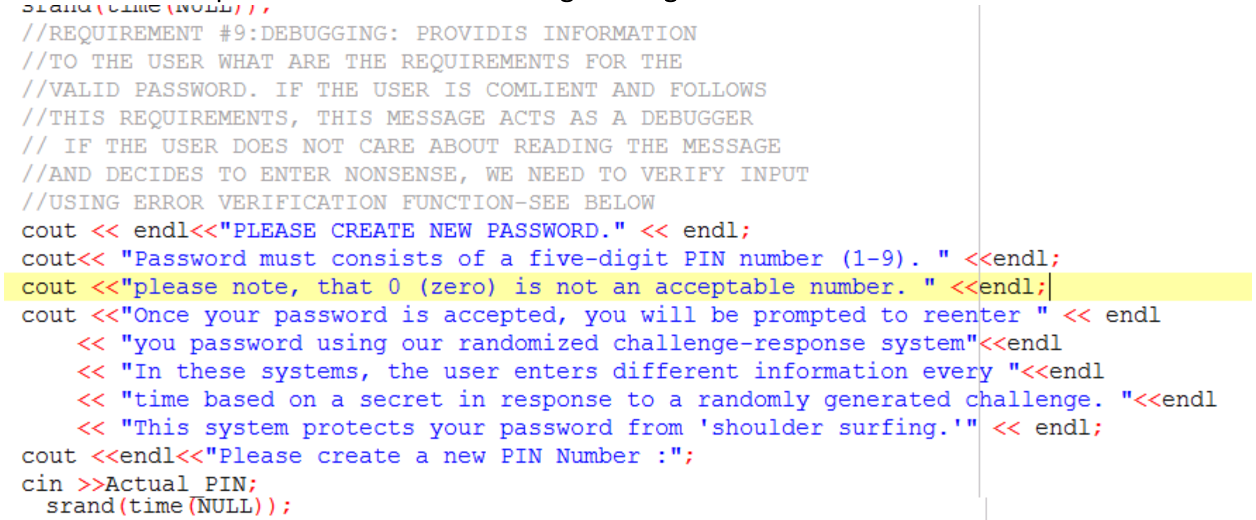


```
//REQUIREMENT #4:Demonstrates Conditional Switch
switch(input)
{
case 1:
    setPassword();
    break;
case 2:
    selectBasket();
    break;
case 3:
    cout << "Rate quality of the produce in the previouse purchases." ;
    reviewGoods();
    break;
case 4:
    orderSeeds();
    break;
case 5:
    cout << "Have a nice day.\n";
    break;
default:
    //this will reset back to menu
    //REQUIREMENT #9: THIS WILL PREVENT RUN TIME ERROR
    //LETS THE USER KNOW THAT INCORRECT ENTRY WAS MADE
    cout << "Error, please pick 1, 2, 3, 4 or 5." << endl;
    menu();
    break;
    }
```

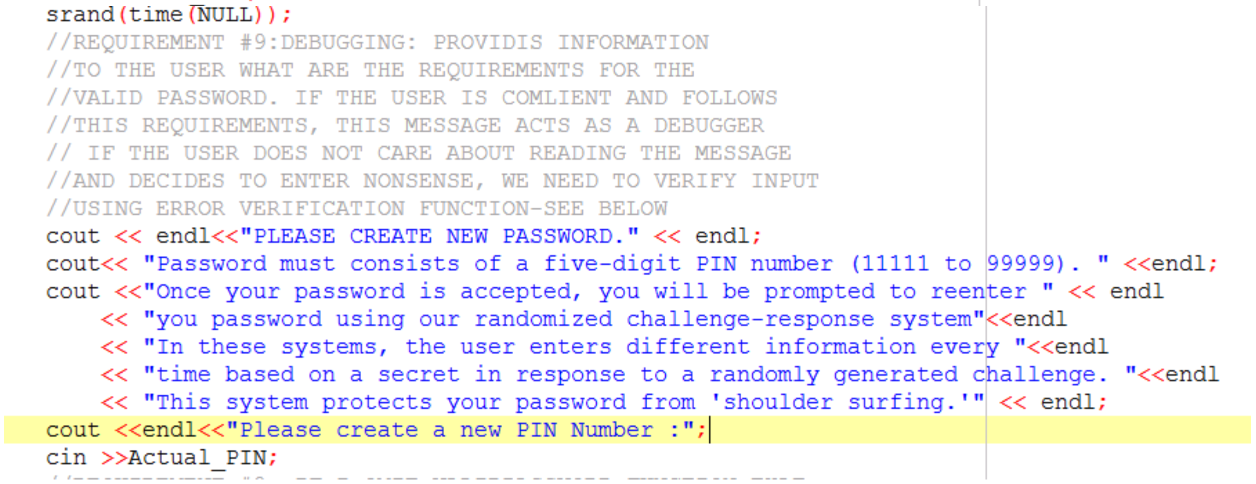
Lets take a look at each subsection closely:

SETPASSWORD:

The user will be provided with the following message:

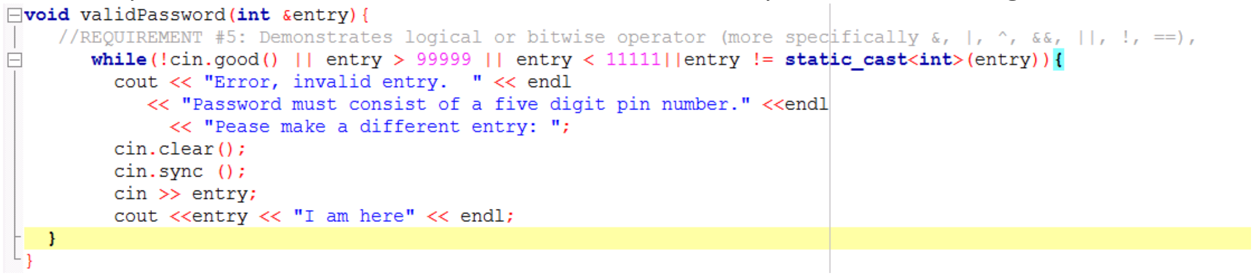


```
srand(time(NULL));
//REQUIREMENT #9:DEBUGGING: PROVIDIS INFORMATION
//TO THE USER WHAT ARE THE REQUIREMENTS FOR THE
//VALID PASSWORD. IF THE USER IS COMLIENT AND FOLLOWS
//THIS REQUIREMENTS, THIS MESSAGE ACTS AS A DEBUGGER
// IF THE USER DOES NOT CARE ABOUT READING THE MESSAGE
//AND DECIDES TO ENTER NONSENSE, WE NEED TO VERIFY INPUT
//USING ERROR VERIFICATION FUNCTION-SEE BELOW
cout << endl<<"PLEASE CREATE NEW PASSWORD." << endl;
cout<< "Password must consists of a five-digit PIN number (1-9). " <<endl;
cout <<"please note, that 0 (zero) is not an acceptable number. " <<endl;
cout <<"Once your password is accepted, you will be prompted to reenter " << endl
    << "you password using our randomized challenge-response system"<<endl
    << "In these systems, the user enters different information every "<<endl
    << "time based on a secret in response to a randomly generated challenge. "<<endl
    << "This system protects your password from 'shoulder surfing.'" << endl;
cout <<endl<<"Please create a new PIN Number :";
cin >>Actual_PIN;
```



```
srand(time(NULL));
//REQUIREMENT #9:DEBUGGING: PROVIDIS INFORMATION
//TO THE USER WHAT ARE THE REQUIREMENTS FOR THE
//VALID PASSWORD. IF THE USER IS COMLIENT AND FOLLOWS
//THIS REQUIREMENTS, THIS MESSAGE ACTS AS A DEBUGGER
// IF THE USER DOES NOT CARE ABOUT READING THE MESSAGE
//AND DECIDES TO ENTER NONSENSE, WE NEED TO VERIFY INPUT
//USING ERROR VERIFICATION FUNCTION-SEE BELOW
cout << endl<<"PLEASE CREATE NEW PASSWORD." << endl;
cout<< "Password must consists of a five-digit PIN number (11111 to 99999). " <<endl;
cout <<"Once your password is accepted, you will be prompted to reenter " << endl
    << "you password using our randomized challenge-response system"<<endl
    << "In these systems, the user enters different information every "<<endl
    << "time based on a secret in response to a randomly generated challenge. "<<endl
    << "This system protects your password from 'shoulder surfing.'" << endl;
cout <<endl<<"Please create a new PIN Number :";
cin >>Actual_PIN;
```

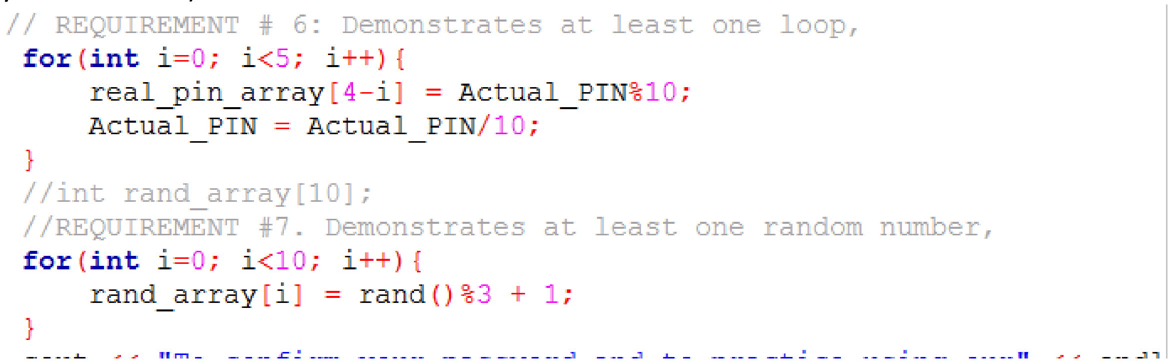
Once the password is entered, it will be checked for validity with the following code:



```
void validPassword(int &entry){
    //REQUIREMENT #5: Demonstrates logical or bitwise operator (more specifically &, |, ^, &&, ||, !, ==),
      while(!cin.good() || entry > 99999 || entry < 11111||entry != static_cast<int>(entry)){
        cout << "Error, invalid entry.  " << endl
           << "Password must consist of a five digit pin number." <<endl
             << "Pease make a different entry: ";
        cin.clear();
        cin.sync ();
        cin >> entry;
        cout <<entry << "I am here" << endl;
   }
}
```

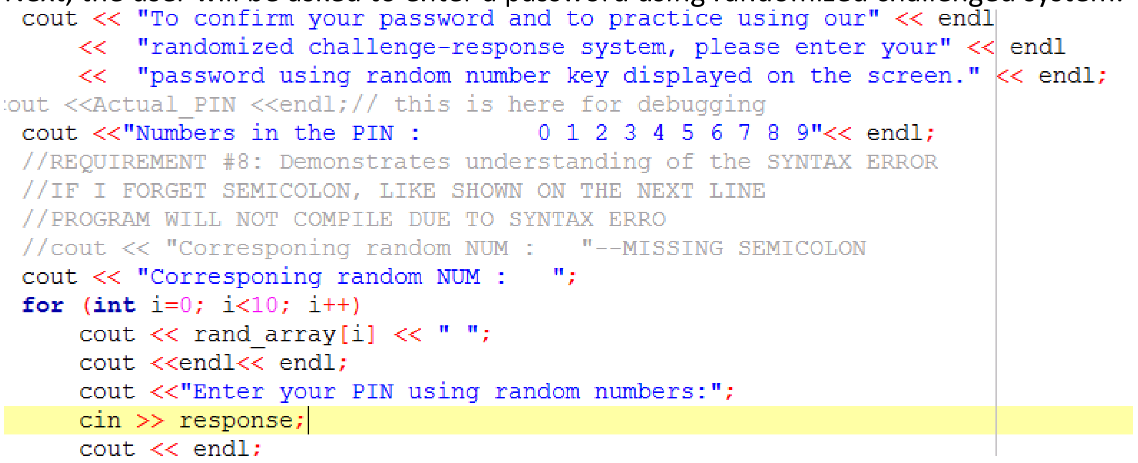
Letters, doubles, and passwords that are not between 11111-99999 will be rejected. Initially, I was planning to accept passwords like 00000, however, I was having a hard time with zeroes and making sure that password is 5 characters long, so 0 will not be acceptable number for password.

Once the password is accepted, 2 different arrays will be generated. One array will be filled with the password that was entered by the user and the second one will be filled with randomly generated numbers (I will use only numbers 1-3):



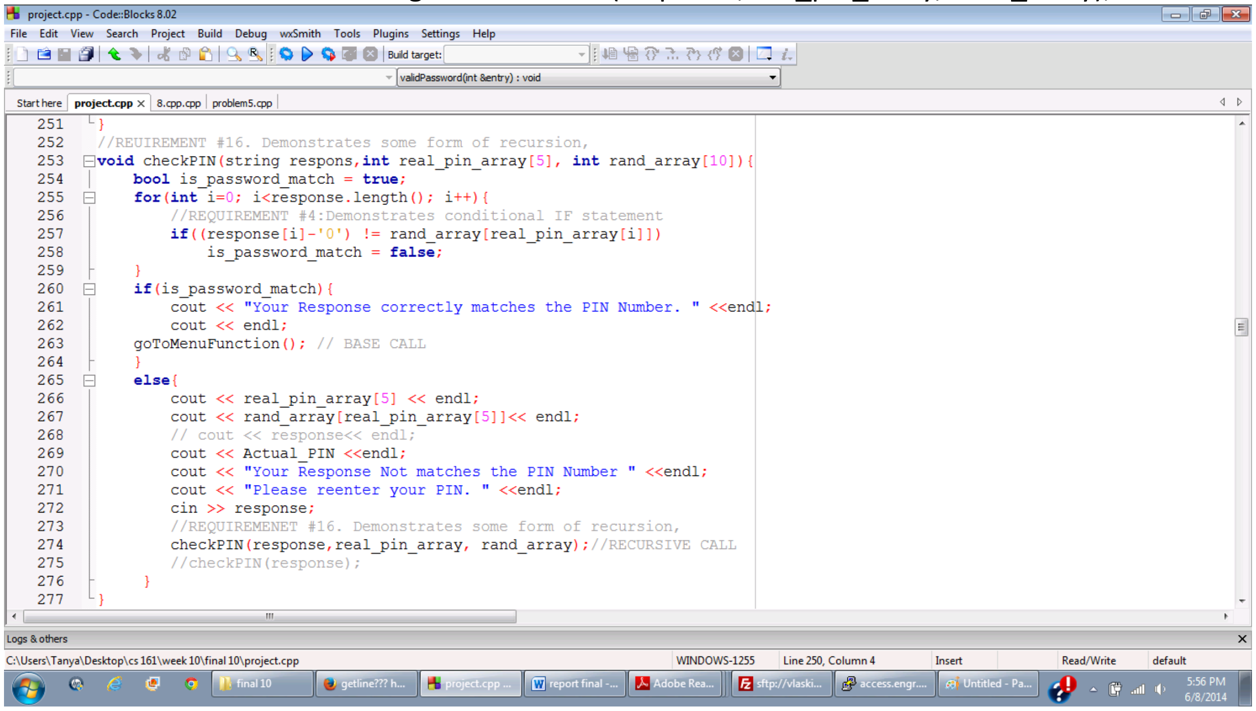
```
// REQUIREMENT # 6: Demonstrates at least one loop,
 for(int i=0; i<5; i++){
     real_pin_array[4-i] = Actual_PIN%10;
     Actual_PIN = Actual_PIN/10;
 }
 //int rand_array[10];
 //REQUIREMENT #7. Demonstrates at least one random number,
 for(int i=0; i<10; i++){
     rand_array[i] = rand()%3 + 1;
 }
```

Next, the user will be asked to enter a password using randomized challenged system:



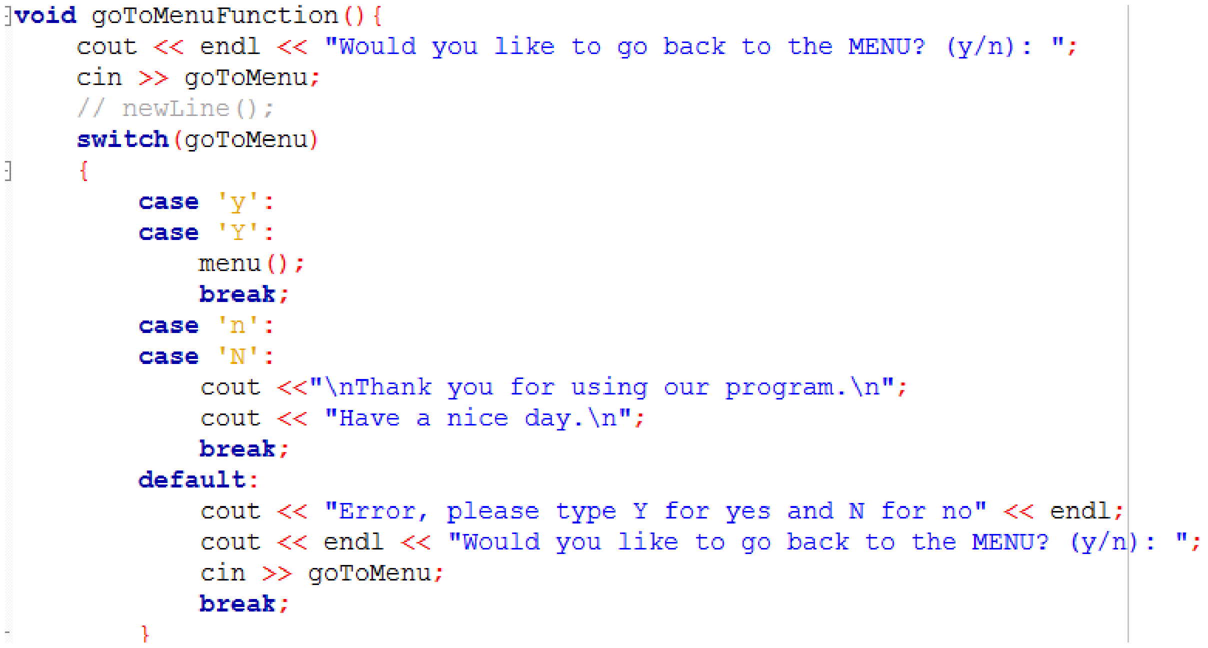
```
 cout << "To confirm your password and to practice using our" << endl
     <<  "randomized challenge-response system, please enter your" << endl
     <<  "password using random number key displayed on the screen." << endl;
out <<Actual_PIN <<endl;// this is here for debugging
 cout <<"Numbers in the PIN :        0 1 2 3 4 5 6 7 8 9"<< endl;
 //REQUIREMENT #8: Demonstrates understanding of the SYNTAX ERROR
 //IF I FORGET SEMICOLON, LIKE SHOWN ON THE NEXT LINE
 //PROGRAM WILL NOT COMPILE DUE TO SYNTAX ERRO
 //cout << "Corresponing random NUM :    "--MISSING SEMICOLON
 cout << "Corresponing random NUM :    ";
 for (int i=0; i<10; i++)
     cout << rand_array[i] << " ";
     cout <<endl<< endl;
     cout <<"Enter your PIN using random numbers:";
     cin >> response;
     cout << endl;
```

Passwords will be checked using the: checkPIN(response,real_pin_array, rand_array); function.



```
}
//REUIREMENT #16. Demonstrates some form of recursion,
void checkPIN(string respons,int real_pin_array[5], int rand_array[10]){
    bool is_password_match = true;
    for(int i=0; i<response.length(); i++){
        //REQUIREMENT #4:Demonstrates conditional IF statement
        if((response[i]-'0') != rand_array[real_pin_array[i]])
            is_password_match = false;
    }
    if(is_password_match){
        cout << "Your Response correctly matches the PIN Number. " <<endl;
        cout << endl;
    goToMenuFunction(); // BASE CALL
    }
    else{
        cout << real_pin_array[5] << endl;
        cout << rand_array[real_pin_array[5]]<< endl;
        // cout << response<< endl;
        cout << Actual_PIN <<endl;
        cout << "Your Response Not matches the PIN Number " <<endl;
        cout << "Please reenter your PIN. " <<endl;
        cin >> response;
        //REQUIREMENET #16. Demonstrates some form of recursion,
        checkPIN(response,real_pin_array, rand_array);//RECURSIVE CALL
        //checkPIN(response);
     }
}
```

This function will check if the entry that was made by the randomized challenge system matches the password that was entered. If the password does not match, the user will be asked to reenter the password. If the password match, the user will get a message that the password matches and will be asked if they want to go to the main menu:

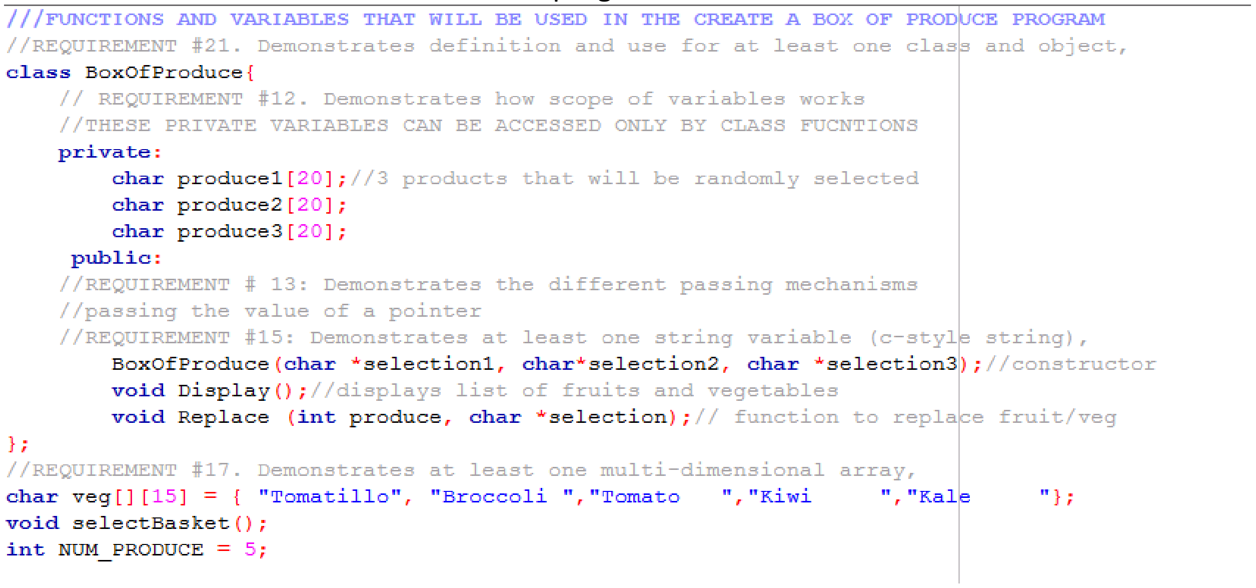


```
void goToMenuFunction(){
    cout << endl << "Would you like to go back to the MENU? (y/n): ";
    cin >> goToMenu;
    // newLine();
    switch(goToMenu)
    {
        case 'y':
        case 'Y':
            menu();
            break;
        case 'n':
        case 'N':
            cout <<"\nThank you for using our program.\n";
            cout << "Have a nice day.\n";
            break;
        default:
            cout << "Error, please type Y for yes and N for no" << endl;
            cout << endl << "Would you like to go back to the MENU? (y/n): ";
            cin >> goToMenu;
            break;
        }
```

Please note that this function is not 100% perfect, so please entry y or n. I did not have time to check it thoroughly and did not have time to improve it to check for valid entry, so please enter only y or n.

CREATE BOX OF PRODUCE OPTION:

List of functions and variables used in this program:



```
///FUNCTIONS AND VARIABLES THAT WILL BE USED IN THE CREATE A BOX OF PRODUCE PROGRAM
//REQUIREMENT #21. Demonstrates definition and use for at least one class and object,
class BoxOfProduce{
    // REQUIREMENT #12. Demonstrates how scope of variables works
    //THESE PRIVATE VARIABLES CAN BE ACCESSED ONLY BY CLASS FUCNTIONS
    private:
        char produce1[20];//3 products that will be randomly selected
        char produce2[20];
        char produce3[20];
     public:
    //REQUIREMENT # 13: Demonstrates the different passing mechanisms
    //passing the value of a pointer
    //REQUIREMENT #15: Demonstrates at least one string variable (c-style string),
        BoxOfProduce(char *selection1, char*selection2, char *selection3);//constructor
        void Display();//displays list of fruits and vegetables
        void Replace (int produce, char *selection);// function to replace fruit/veg
};
//REQUIREMENT #17. Demonstrates at least one multi-dimensional array,
char veg[][15] = { "Tomatillo", "Broccoli ","Tomato   ","Kiwi     ","Kale     "};
void selectBasket();
int NUM_PRODUCE = 5;
```

I decided to demonstrate my understanding of classes, and multidimensional arrays.

First step is to generate basket randomly. The basket will contain one of the products listed in the array, duplicate products will be accepted (example: tomato, kiwi and tomato) will be an acceptable basket.

```
BoxOfProduce box(veg[rand()%5], veg[rand()%5], veg[rand()%5] );
```

Please note that this is a constructor with parameters passed as references.

```
BoxOfProduce::BoxOfProduce(char *selection1, char*selection2, char *selection3){
   strcpy(produce1, selection1);
   strcpy(produce2, selection2);
   strcpy(produce3, selection3);
}
```

Next the contents of the basket will be displayed to the user and the user will be asked if they want to replace product in the basket.

```
void BoxOfProduce::Display(){
   cout<<"1. "<<produce1<<endl;
   cout<<"2. "<<produce2<<endl;
   cout<<"3. "<<produce3<<endl;
}
```

If the user decides that they want to replace something in the basket the following function will be called:

```
void BoxOfProduce::Replace (int choice, char *selection){
   switch(choice){
      case 1: strcpy(produce1, selection); break;
      case 2: strcpy(produce2, selection); break;
      case 3: strcpy(produce3, selection); break;
   }
}
```

The function will be called in the following way:

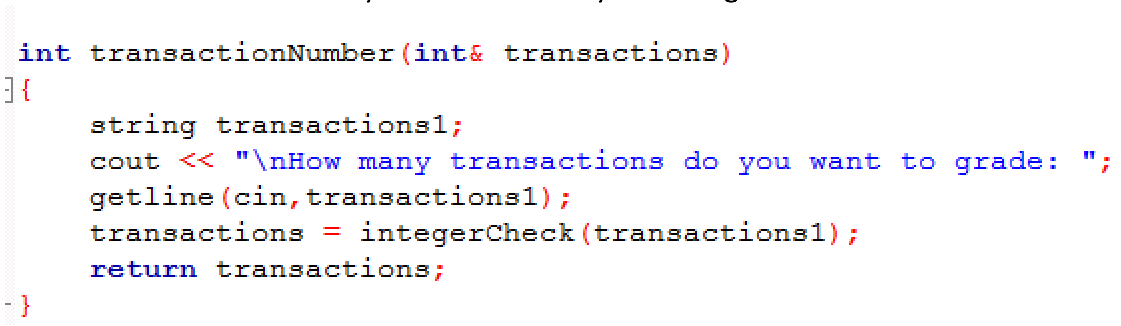
```
            cout<<"\nChose the index of the frut or vegetable that you want to choose : ";
           getline(cin,rep_with1);
            rep_with = integerCheck(rep_with1);
```

box.Replace(rep, veg[rep_with-1]);

If the user does not want to replace any good, they will be asked if they want to go back to the main menu.

RATE QUALITY OF THE PRODUCE IN THE PREVIOUS TRANSACTIONS:

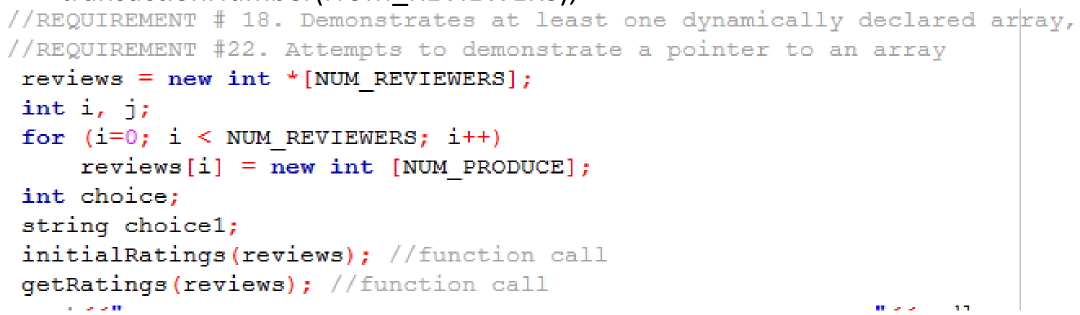
First user is asked how many transactions they want to grade:



```
int transactionNumber(int& transactions)
{
    string transactions1;
    cout << "\nHow many transactions do you want to grade: ";
    getline(cin,transactions1);
    transactions = integerCheck(transactions1);
    return transactions;
}
```

Next a dynamic array is generated. This array will hold review of each product in each transaction.

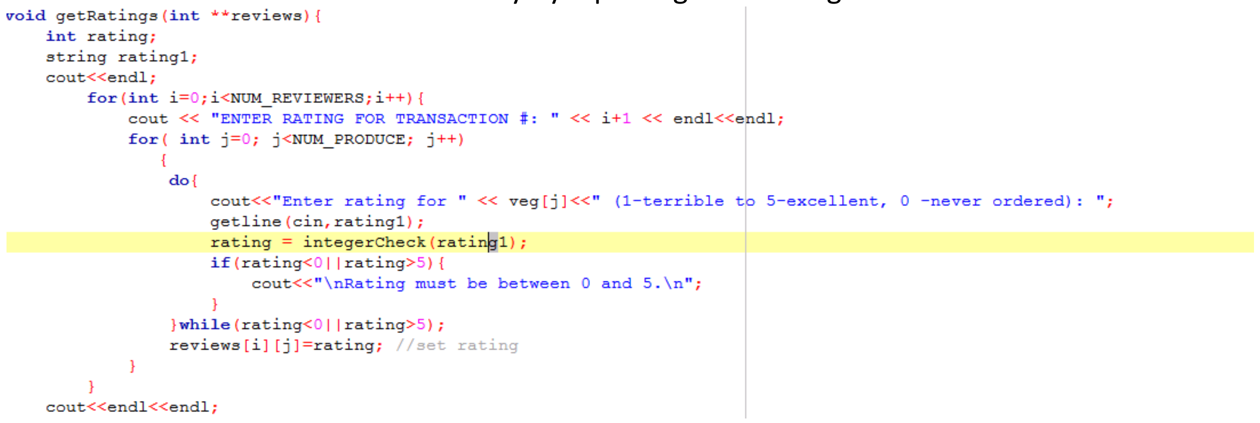
transactionNumber(NUM_REVIEWERS);



```
//REQUIREMENT # 18. Demonstrates at least one dynamically declared array,
//REQUIREMENT #22. Attempts to demonstrate a pointer to an array
 reviews = new int *[NUM_REVIEWERS];
 int i, j;
 for (i=0; i < NUM_REVIEWERS; i++)
     reviews[i] = new int [NUM_PRODUCE];
 int choice;
 string choice1;
 initialRatings(reviews); //function call
 getRatings(reviews); //function call
```

Next step is initialize rating to 0, by using initialRatings function.

Next is to ask the user to fill in the array by inputting their rating scores.



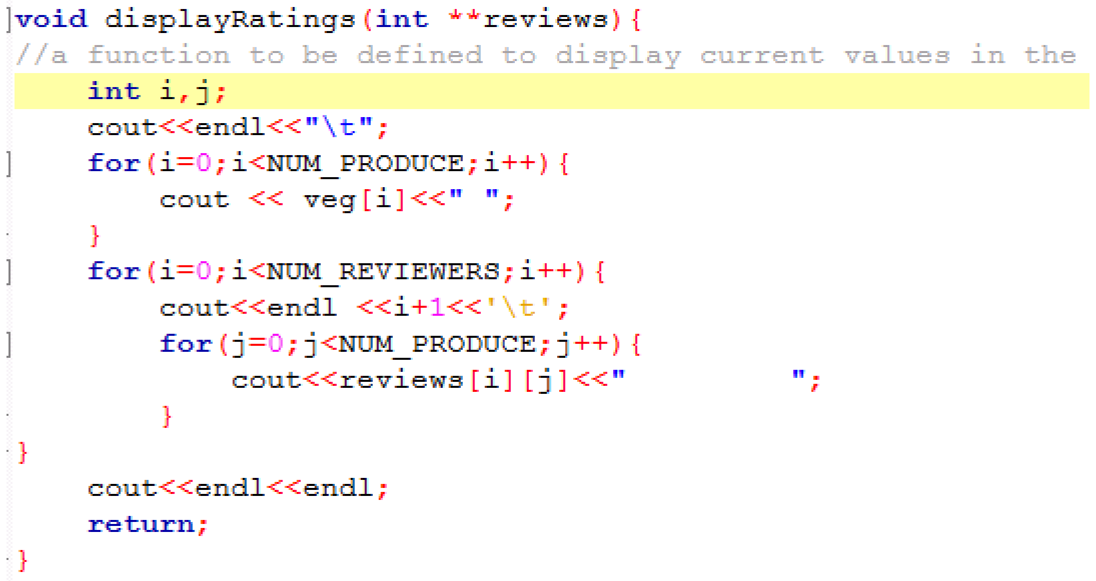
```
void getRatings(int **reviews){
    int rating;
    string rating1;
    cout<<endl;
        for(int i=0;i<NUM_REVIEWERS;i++){
            cout << "ENTER RATING FOR TRANSACTION #: " << i+1 << endl<<endl;
            for( int j=0; j<NUM_PRODUCE; j++)
              {
               do{
                   cout<<"Enter rating for " << veg[j]<<" (1-terrible to 5-excellent, 0 -never ordered): ";
                   getline(cin,rating1);
                   rating = integerCheck(rating1);
                   if(rating<0||rating>5){
                       cout<<"\nRating must be between 0 and 5.\n";
                   }
               }while(rating<0||rating>5);
               reviews[i][j]=rating; //set rating
            }
        }
    cout<<endl<<endl;
```

One the user makes their entries; they are asked what they want to do with the entries using the following menu:

```
 switch (choice)
 {
      case 1:
           displayRatings(reviews);break;
      case 2:
           showAverageRatings(reviews); break;
      case 3:
           menu(); break;
      default:
           cout<<"You enter an invaild choice.\n";break;
      }
     delete2D_ArrayPointer(reviews);
```

I think these functions are self-explanatory.

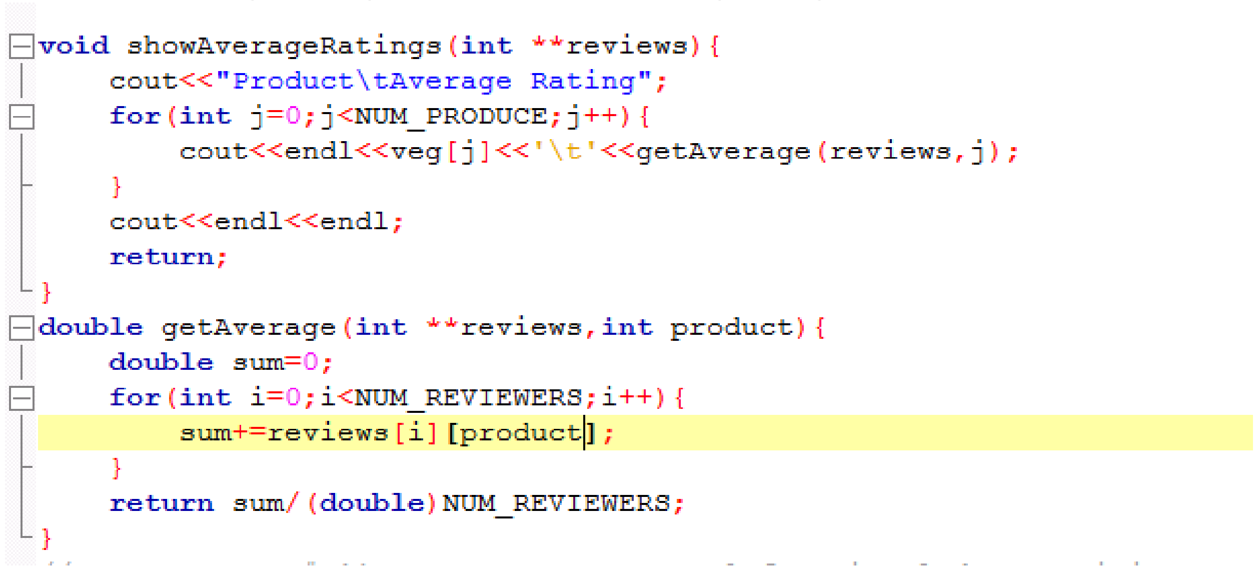
Display rating is a simple functions that takes 2D array and displays it to the user:



```
void displayRatings(int **reviews){
//a function to be defined to display current values in the
     int i,j;
     cout<<endl<<"\t";
     for(i=0;i<NUM_PRODUCE;i++){
          cout << veg[i]<<"  ";
     }
     for(i=0;i<NUM_REVIEWERS;i++){
          cout<<endl <<i+1<<'\t';
          for(j=0;j<NUM_PRODUCE;j++){
               cout<<reviews[i][j]<<"            ";
          }
}
     cout<<endl<<endl;
     return;
}
```

Show average rating is a function that shows average rating for each product in the basket.

This function uses getAverage function calculate average rating.

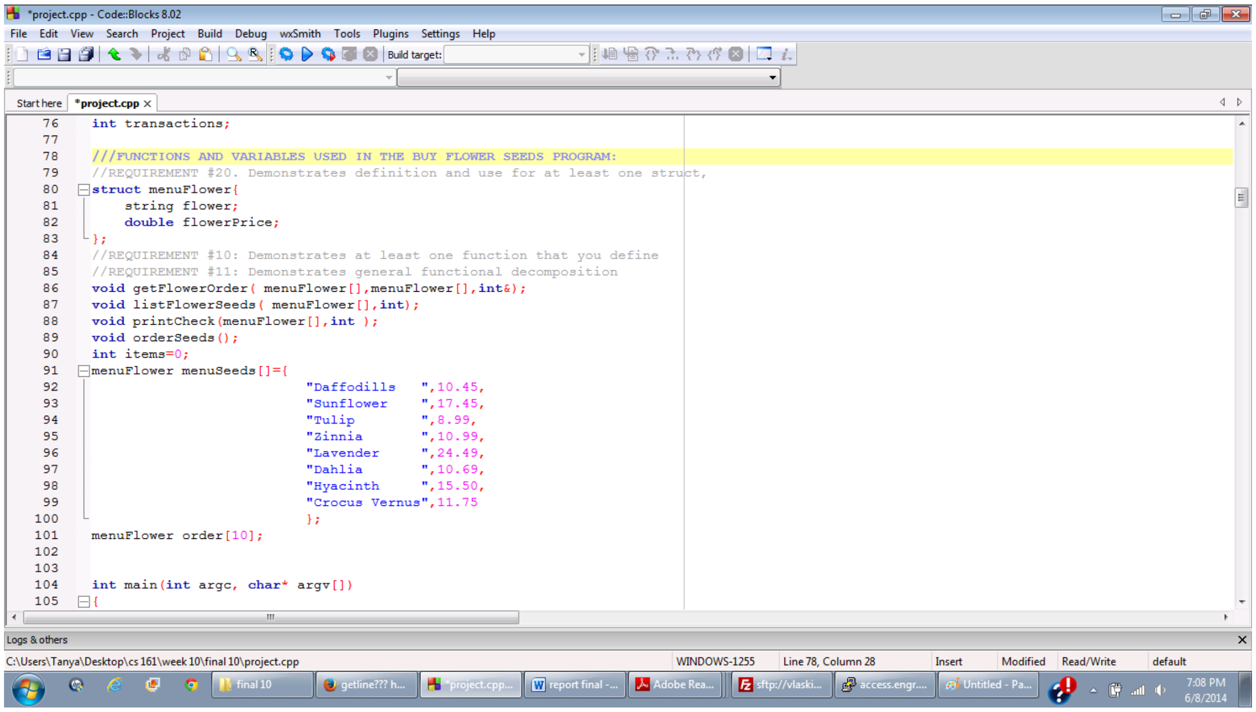


```
void showAverageRatings(int **reviews){
     cout<<"Product\tAverage Rating";
     for(int j=0;j<NUM_PRODUCE;j++){
          cout<<endl<<veg[j]<<'\t'<<getAverage(reviews,j);
     }
     cout<<endl<<endl;
     return;
}
double getAverage(int **reviews,int product){
     double sum=0;
     for(int i=0;i<NUM_REVIEWERS;i++){
          sum+=reviews[i][product];
     }
     return sum/(double)NUM_REVIEWERS;
}
```

Another thing that I almost forgot do to is to delete pointers to 2D dynamic array. I wrote the following function. I hope it does the job:

```
//Deletes pointer to 2d array
void delete2D_ArrayPointer(int **reviews){
    for(int i=0;i<NUM_REVIEWERS;i++){
        for(int j=0;j<NUM_PRODUCE;j++){
            //delete reviews[i][j]; //set tc
            delete[]reviews[i];
        }
    }
```

ORDER SEEDS:



```
*project.cpp - Code::Blocks 8.02
```

```
int transactions;

///FUNCTIONS AND VARIABLES USED IN THE BUY FLOWER SEEDS PROGRAM:
//REQUIREMENT #20. Demonstrates definition and use for at least one struct,
struct menuFlower{
    string flower;
    double flowerPrice;
};
//REQUIREMENT #10: Demonstrates at least one function that you define
//REQUIREMENT #11: Demonstrates general functional decomposition
void getFlowerOrder( menuFlower[],menuFlower[],int&);
void listFlowerSeeds( menuFlower[],int);
void printCheck(menuFlower[],int );
void orderSeeds();
int items=0;
menuFlower menuSeeds[]={
                                "Daffodills   ",10.45,
                                "Sunflower    ",17.45,
                                "Tulip        ",8.99,
                                "Zinnia       ",10.99,
                                "Lavender     ",24.49,
                                "Dahlia       ",10.69,
                                "Hyacinth     ",15.50,
                                "Crocus Vernus",11.75
                                };
menuFlower order[10];


int main(int argc, char* argv[])
{
```

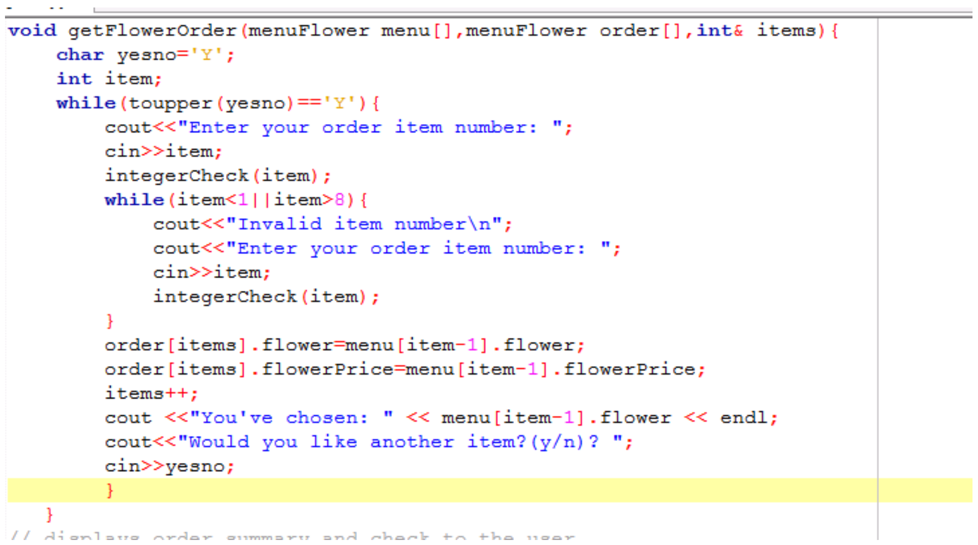
```
C:\Users\Tanya\Desktop\cs 161\week 10\final 10\project.cpp    WINDOWS-1255    Line 78, Column 28    Insert    Modified    Read/Write    default
```

Ill use structures for this program.

First user will be presented with the list of flower seeds that they can buy:

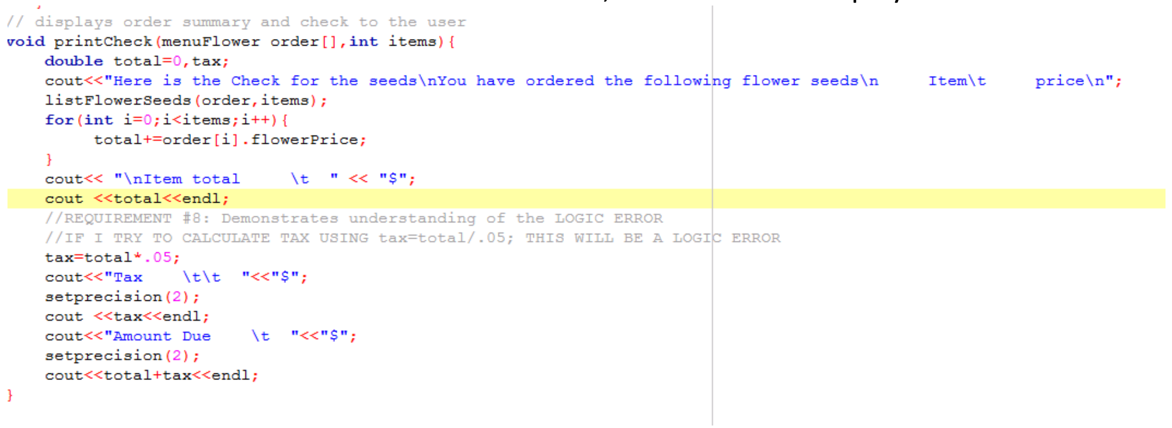
```
//displays list of flower seeds
//REQUIREMENT #23. Attempts to demonstrate a pointer to a struct (could be a bit tough),
void listFlowerSeeds(menuFlower flowers[],int n){
    menuFlower *pointers[n];
    for( int i=0;i<n;i++){
        pointers[i]=&flowers[i];
        cout<<i+1<<". "<<pointers[i]->flower<<"  $";
        cout<<pointers[i]->flowerPrice<<endl;
    }
}
```

Next the user will be asked what they want to order:



```
void getFlowerOrder(menuFlower menu[],menuFlower order[],int& items){
    char yesno='Y';
    int item;
    while(toupper(yesno)=='Y'){
        cout<<"Enter your order item number: ";
        cin>>item;
        integerCheck(item);
        while(item<1||item>8){
            cout<<"Invalid item number\n";
            cout<<"Enter your order item number: ";
            cin>>item;
            integerCheck(item);
        }
        order[items].flower=menu[item-1].flower;
        order[items].flowerPrice=menu[item-1].flowerPrice;
        items++;
        cout <<"You've chosen: " << menu[item-1].flower << endl;
        cout<<"Would you like another item?(y/n)? ";
        cin>>yesno;
        }
    }
```

If the user does not want to order more items, a check will be displayed:

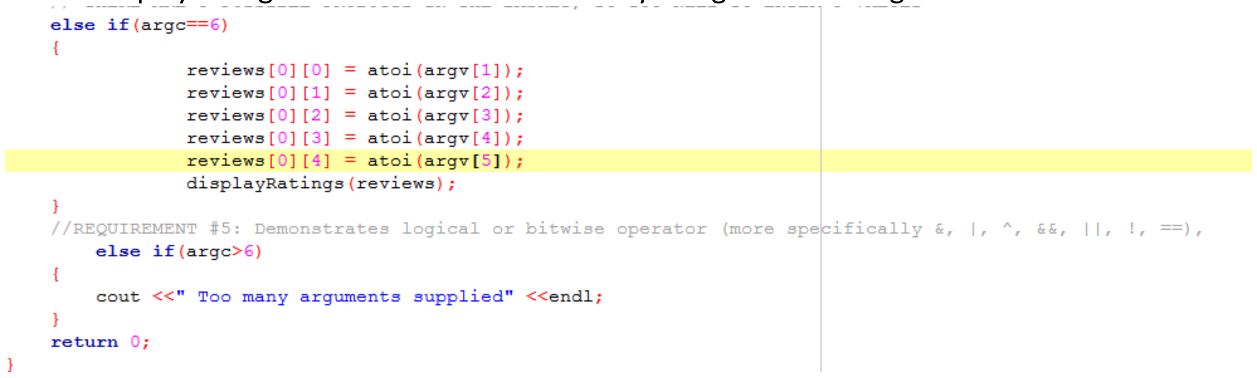


```
// displays order summary and check to the user
void printCheck(menuFlower order[],int items){
    double total=0,tax;
    cout<<"Here is the Check for the seeds\nYou have ordered the following flower seeds\n     Item\t     price\n";
    listFlowerSeeds(order,items);
    for(int i=0;i<items;i++){
        total+=order[i].flowerPrice;
    }
    cout<< "\nItem total     \t  " << "$";
    cout <<total<<endl;
    //REQUIREMENT #8: Demonstrates understanding of the LOGIC ERROR
    //IF I TRY TO CALCULATE TAX USING tax=total/.05; THIS WILL BE A LOGIC ERROR
    tax=total*.05;
    cout<<"Tax    \t\t  "<<"$";
    setprecision(2);
    cout <<tax<<endl;
    cout<<"Amount Due    \t  "<<"$";
    setprecision(2);
    cout<<total+tax<<endl;
}
```

Precision will be placed to 2 using the following function:

```
void setprecision(int number){
   cout.setf(ios::fixed);
   cout.setf(ios::showpoint);
   cout.precision(number);
}
```

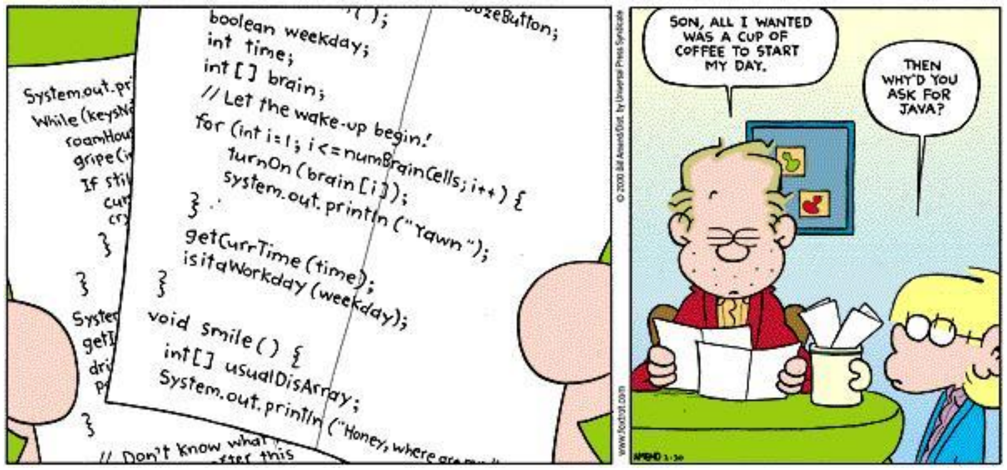
It seems that I met most of the requirements of the project; the last one is command line input. My plan was to use displayRating function for 1 transaction by using the following code.



```
    else if(argc==6)
    {
                reviews[0][0] = atoi(argv[1]);
                reviews[0][1] = atoi(argv[2]);
                reviews[0][2] = atoi(argv[3]);
                reviews[0][3] = atoi(argv[4]);
                reviews[0][4] = atoi(argv[5]);
                displayRatings(reviews);
    }
    //REQUIREMENT #5: Demonstrates logical or bitwise operator (more specifically &, |, ^, &&, ||, !, ==),
        else if(argc>6)
    {
        cout <<" Too many arguments supplied" <<endl;
    }
    return 0;
}
```

The idea was that if the user enters 5 numbers, one for each product, I should be able to use displayRating functions to see the ratings that were entered. However, I keep getting Segmentation fault (core dumped error.) I do not have time to troubleshoot it right now, so I'll leave it as it is. I have not met requirement 19. I do not even understand why anyone would prefer command line over regular cin/cout. You do not even know what you are doing/entering in command line, while when you use cin/cout you can follow directions displayed on the screen.
THANKS FOR GRADING MY ASSIGNMENTS.

—You're fluent in twenty-four programming languages, but you can't even talk about the weather with me!"
Igor Aleshin
DATAART Enjoy IT


```
boolean weekday;
int time;
int [] brain;
// Let the wake-up begin!
for (int i=1; i<=numBrainCells; i++) {
  turnOn (brain[i]);
  System.out.println ("Yawn");
}
getCurrTime (time);
isitaWorkday (weekday);
}
void smile() {
int[] usualDisArray;
System.out.println ("Honey, where are
```
SON, ALL I WANTED WAS A CUP OF COFFEE TO START MY DAY.
THEN WHY'D YOU ASK FOR JAVA?