WEEK 7 REPORT
Tatyana Vlaskin

PROBLEM 1:

The problem asks us to write a function that creates a variable of some data type, create a pointer that points to that variable, ask the user for a value to assign to that variable, and displays variable and address of the variable using different methods. I've also decided to see what will happen when we try to display the address of the point and it seems that it worked. The problem is straightforward and if you look at the implementation, you will be reading an implementation. The only thing that I want to point out, when I initially wrote the program, I decided to put everything is the main method. So I program looked like that:

```
int main(){

        int variable; // creating a variable of int data type
        int *pointer = &variable; //creating a pointer that points to the variable

        cout << "Enter an integer: ";
        cin >> variable;
        validEntry(variable); // making sure that user entered int type
        cout << endl << "1. To access the value of a variable directly, type: " << endl;
        cout << "cout << variable." << endl;
        cout << "You will get: " << variable << endl;

        cout << endl << "2. To access the address of a variable with the reference operator, type: " << endl;
        cout << "cout << &variable." << endl;
        cout << "You will get: " << &variable << endl;

        cout << endl << "3. To access a pointer's value by accessing it directly, type: " << endl;
        cout << "cout << pointer." << endl;
        cout << "This is equivalent to accessing the address of the variable with the pointer." << endl;
        cout << "You will get: " << pointer << endl;

        cout << endl << "4. To access the value of the pointee by dereferencing a pointer, type: " << endl;
        cout << "cout << *pointer." << endl;
        cout << "This is equivalent to accessing the value of a variable." << endl;
        cout << "You will get: " << *pointer << endl;

        cout << endl << "5. To access the address of a pointer, type: " << endl;
        cout << "cout << &pointer." << endl;
        cout << "You will get: " << &pointer << endl;
```

However, I decided that that was too much for the main method and rewrote the program using 5 different functions. So right now, my main method looks like this:

```
int main(){

        int variable; // creating a variable of int data type
        int *pointer = &variable; //creating a pointer that points to the variable

        cout << "Enter an integer: ";
        cin >> variable;
        validEntry(variable); // making sure that user entered int type

        directVariable(variable); // accessing variable virectly

        referenceOperator(variable);// accessing address using reference operator

        pointerValue (pointer); // accesssing address using pointer

        valuePointeeDereference(pointer); // accessing value using pointer and dereference operator

        pointerAddress(pointer); // accessing address of a pointer

}
```

PROBLEM 2:

We need to write a function that:

1. creates an array of int values dynamically (with the new keyword)
2. asks the user to input values to fill it
3. prints the largest and smallest values stored in the array
4. if the user makes invalid entry, lets the use know that invalid entry was made.

So we need to declare dynamic array by doing something like that:
The use will be asks how many numbers they want to enter: arraySize;
After that an array will be declaredas: int *inputArray = new int[arraySize];
After that Ill ask user to enter integers and check that the valid entry was made.
Next, will have fill the array function. As the user keeps entering number till all arraySize numbers are enteres, those numbers will go into the array:

```
void fillArray( int array[], int size)
{
   for ( int index = 0; index < arraySize; index++){
   cout << "Enter an integer " << index +1 << ": ";
   cin >> array[index];
   validEntry(array[index], "Enter an integer: ");
   }
}
```

Once all the numbers are entered the array will be sorter. These is an example in the book that shows us how to sort array, but after looking for long time, I decided that its too complex. I am providing it for completeness and mainly because I use this report to study for the test:

```
34 void sort( int a[], int numberUsed)
35 {
36 int indexOfNextSmallest;
37 for ( int index = 0; index < numberUsed – 1; index++)
38 { //Place the correct value in a[index] :
39 indexOfNextSmallest =
40 indexOfSmallest(a, index, numberUsed);
41 swapValues(a[index], a[indexOfNextSmallest]);
42 //a[0] <= a[1] <= ... <= a[index] are the smallest of the
43 //original array elements. The rest of the
//elements are in the remaining positions .
44 }
45 }
```

Instead, I came up with the following sort function:

```
void sortArray(int array[], int size)
{
   for (int i = 0; i < size -1; i++){
      for (int j = i+1; j <size; j++){
         if (array[i] > array[j]){
            swap(array[i], array[j]);
         }
      }
   }
}
```

Where swap is the function that I got from the book and modified it by adding pointers.

```
void swapValues( int *number1, int *number2)
 {
   int temp = *number1;
   *number1 = *number2;
   *number2 = temp;
 }
```

Finally inputArray[0] and inputArray[arraySize-1] values will be displayed as the smallerst and largest numbers respectively.

PROBLEM 3:

We need to write a program that does the following:

1. Asks the user how many students took the quiz.
2. Creates dynamic array: int *entry = new int[students];
3. Allows the user to enter grades for each student
4. As the grades are being entered, they are stored in the dynamic array and
5. At the same time each grade as counted once its entered
6. If the user makes invalid entry, the user is notified that entry was invalid and that entry does not go to the array.

To do this problem I took my code from last week and have done few modifications.
I created a dynamic array: int *entry = new int[students] to store entries made by the user.
Next, I decided to create fill the grade function, which is filled as the user enters grades:

```
void fillGrades( int array[], int size)
{
   for ( int index = 0; index < students; index++){
   cout << "Enter students " << index +1 << " grade: ";
```

```
   cin >> array[index];
   gradeCheck(array[index]);

}
```
And finally, I had to change last weeks function :
```
void countGrade(int grades[], int grade)
{
   if (grade == 0)
      grades[0]++;
   else if (grade == 1)
      grades[1]++;
   else if (grade == 2)
      grades[2]++;
   else if (grade == 3)
      grades[3]++;
   else if (grade == 4)
      grades[4]++;
   else if (grade == 5)
      grades[5]++;
}
```
To

```
void countGrade(int grades[], int entry[])
{
         Int k =0;
   if (entry[k] == 0)
      grades[0]++;
   else if (entry[k] == 1)
      grades[1]++;
   else if (entry[k] == 2)
      grades[2]++;
   else if (entry[k] == 3)
      grades[3]++;
   else if (entry[k]== 4)
      grades[4]++;
   else if (entry[k] == 5)
      grades[5]++;
}
```
The program compiled and ran, but I grades were not counted. I had to get help on discussion board and Brian Tiegs told me that my problem was that I never set a value for k in my

countgrades function. I could either pass the index variable from your for loop in int main() to the function and set k equal to index. Or I cound move the for loop into the function. For this I would need to pass in your students variable.
Following his advice my function was changed to:

```
void countGrade(int grades[], int entry[], int k)
{
   //int k;
   if (entry[k] == 0)
      grades[0]++;
   else if (entry[k] == 1)
      grades[1]++;
   else if (entry[k] == 2)
      grades[2]++;
   else if (entry[k] == 3)
      grades[3]++;
   else if (entry[k]== 4)
      grades[4]++;
   else if (entry[k] == 5)
      grades[5]++;
}
```

And that solved the problem.

PROBLEM 4:

For this program, we need to write a tictactoe game using dy mic arrays. We are allowed to choose dimension.

I decided to use 2D dynamic array mainly because I do not think that I understood them completely last week.
I went back to my last week code and have done few modifications.
First I've replayed table[3][3] array with the dynamic array. I was able to do that following instructions in the book that we use in this class.

```
char **table;
table = new char *[rows];
int i, j;
for (i=0; i < rows; i++)
table[i] = new char [columns];
```

That was very straightforward and I was special that that was the end of this problem. I went to discussion board and found that that was exactly what other people did. However, I think every single code that I saw online used 1D array. I've done a better job in testing this program and found out that last week I submitted this program with several bugs.

1. My code last week was:

```
if (!input())    // if enry is invalid, display invalid message
{
   cin.clear();
   while (cin.get() != '\n');
   cout << "Invalid entry! Try one more time" << endl;
}
count++;    //count entries
```

I was incrementing count when the user was making wrong input, like character, number outside of range 1-9, ect. As a results of this the user was getting an invalid entry message but was not allowed to reentry their move . I changed the code ad it fixed the problem. Instead of incrementing, I decrement count when wrong move is made.

```
if (!input())    // if enry is invalid, display invalid message
{
   cin.clear();
   while (cin.get() != '\n');
   cout << "Invalid entry! Try one more time" << endl;
}
count--;    //count entries
```

2. Second bug is the last weeks problem was that incorrect winner was displayed. Its hard to explaine why that was happening, but Ill try.
   So last week I had:

```
if (step){         // display the name of the winner on the screen.
    cout <<"Congratulations! " << PlayerName1 << " win!";}
 else{
    cout <<"Congratulations! " <<PlayerName2<< " win!" <<endl;}
```

   In reality it should have been like that:

```
if (step){         // display the name of the winner on the screen.
    cout <<"Congratulations! " << PlayerName2 << " win!";}
 else{
    cout <<"Congratulations! " <<PlayerName1<< " win!" <<endl;}
```

   Lets say Player1 makes a winning move, at this point step=true. The way how the code is written, once the move is made step changes to false to let the other player make an entry and only after that I check for the winner and if there is no winner, let the other player make an entry and if there is a winner display the name of the winner. So if the winner if step true, when we display results, we need to display results for step false because step was changed to false. I do not think that I am doing a really good explanation here. Its very counterintuitive at least for me.
   I think that was it with fixing the code from last week.
   However, I've encountered some problems this week.

At the beginning of the program, I ask user to enter number of rows on the board. Once the entry is made, I ask the user two enter 2 names of the players. The problem was the program was not letting me to enter the name of the first player, it was skipping cin>>Player1 and was prompting me to enter name of the second user. I had to change my code from:

```
cout << "Enter a name of the 1st player. This player will use 'X' character : ";
cin.getline(PlayerName1,80);
cout << "Enter a name of the 2st player. This player will use 'O' character. " <<
endl;
cout << "Name has to be different from the 1st player: ";
cin.getline(PlayerName2,80);
cout << "\n";
```

to

```
do {
  cout << "Enter a name of the 1st player: ";
  cin >> player1;
  cout << "Enter a name of the 2st player: " << endl;
  cout << "Name has to be different from the 1st player: ";
  cin >> player2;
  cout << "\n";
} while (!strcmp(player1, player2));
```

Basically I replaced cin.getline with cin. I have to admit, I do not understand this. I tried to Try flush the stream with something like cin.ignore() and it did not solve the problem. So I decided to go ahead and replace it with cin.

1D ARRAY tic tac toe.

For extra practice, Ill go ahead and do this problem using 1D array and Ill try to design a program for variable board size, but because of the complexity of the program, player will be able to win only if they fill out column or row completely. There will be no diagonal winner.

1. First of all I'll make one array for the board: char *cells = new char[rows*rows]; or it can be char *cells = new char[rows*columns]; because number of rows will equal number of columns.
2. Next, I will make 4 more dynamic arrays to keep track of X and O on the rows and columns. This is necessary for winning.
    a. int *r1 = new int[rows];// array to count X on each row
    b. int *c1 = new int [columns];// array to count X in each column
    c. int *r2 = new int [rows]; // array to count O on each row
    d. int *c2 = new int [columns];// array to count O in each column

e. Initially I can panning to count X and O, but it did not work. I was not able to win, so I had to rewrite this design.
This was my code for the row check originally:

```
void checkWinner(){
    for (int i = 0; i < rows; i++){ // loop through each row and check what its filled with
        int count1  = 0;// to count X on each row, count is reset at the beggining of each r
        int count2 = 0;// to count O on each row, count is reset at the beggining of each ro

        for (int k = 0; k < columns; k++){
            if ((cells[rows*i+k] == 'X')){
            count1++;
            cout << "Count1 is counting X on each row: " << count1 << endl;//<< cells[d]
          }
        else if( (cells[rows*i+k] == 'O')){
        count2++;
        cout << "Count2 is counting O on each row: " << count2 << endl;// << cells[d]<<
        }
    }
    if(count1==rows){
        break;
```

It was a battle, but it did not work, so I decided to try approach with 4 arrays to store X and O in each column and row.

3. Next, I will ask the user to enter how many rows they want to play.
4. Next, I'll do this trick to assign rows = columns:
    a. int temp;
    b. temp = rows;
    c. columns = temp; // colums is eaqual to rows
5. Next I'll go back to my last weeks code and see what I can use from there. I am hoping to use 80% of the code from last week
6. I will initialize each element in the board array to Z. Just random character.
7. I need to change the way how I display the board to the user. I need to make a nested loop.

```
for (int k = 0; k < rows; k++){// loop the rows
            for (int i = 0; i < columns; i++){// loop the columns
    if ( cells[k*rows + i] == 'Z')
    {
    cout << '|' << "-"<< k * rows + i + 1 << "-" << '|';
    }
            // If it's not a Z however, display whatever letter happens to be stored
there
    else
    {
    cout << '|' << "-"<< cells[k*rows + i]<< "-" << '|';
    }
}
```

I had to waste over 5 pages of paper to come up with this relationship: k * rows + i + 1

8. For check winner function I am planning to do something like this:

Basically it is a simple nested loop (several loops). First I loop at each row and store elements in the array, lets say r1, which is designed to store x, if there are as many X as number of columns, than there is a winner, if not we move one. I do this for every single row and every single column. At the beginning of each row and each column, array is initialized to zero.

```
void checkWinner(){
    for(int i=0;i<rows;i++)
   {
      r1[i]=c1[i]=r2[i]=c2[i]=0;// at the beggining of each row and each column, array it initialized to zero
   }
     //checking rows
   for (int i = 0; i < rows; i++){
      for (int j = 0; j < columns; j++){
         if(cells[i*rows+j]=='X')
            r1[i]++;// increment if you see x
         if(cells[i*rows+j]=='O')
            r2[i]++;// increment if you see o
             }
   }
             //check columns, similar to check for rows
   for (int i = 0; i< columns; i++){
              for (int k = 0; k < rows; k++){
         if ((cells[i+ k*rows] == 'X'))
            c1[i]++;
         if ((cells[i+ k*rows] == 'O'))
            c2[i]++;
             }
   }
   resultWiner();
}
```

9. If there is a winner I change bool result winner and the program should stop, but it does not. However, I spent over 20 hours on this problem already and I will stop. So I tested the program for 2x2, 3x3, 4x4- the board that is displayed to the user looks really bad:

```
|-1-||-2-||-3-||-4-|
|-5-||-6-||-7-||-8-|
|-9-||-10-||-11-||-12-|
|-13-||-14-||-15-||-16-|
```

However, I do not think that Ill be able to check formation because I do not know a fast way of doing that and I do not have time to look for this infor online because I need to start studying for the test or I will fail it.
So the main problem that I have with the problem is that when there is a winner, I do get a message that there is a winner, but the problem keep going. It asks the other user make the move. It does not make sense. If there this a winner, I return true, so I should not be entering :

```
while (!resultWiner()){
```

} loop. But I do. I do not know why. Any feedback is appreciated.

PROBLEM 5:

To do this problem, I read the following blogs ad watched the following video:

1. http://www.intechgrity.com/how-to-write-programs-using-command-line-arguments-in-a-few-easy-steps/
2. http://www.youtube.com/watch?v=3iATEcmhB8w
3. http://www.learncpp.com/cpp-tutorial/713-command-line-arguments/
4. http://www.site.uottawa.ca/~lucia/courses/2131-05/labs/Lab3/CommandLineArguments.html

Main method will be like that: int main (int argc, char *argv[])

The argc counts the number of arguments and argv is an array of char pointers which will contain the list of arguments that the user will enter.

Next I'll do a for loop : for(int entry =1; entry <argc-1; entry++){

grades[entry-1]=atoi(argv[entry])}

The function "atoi" discards any whitespace characters until first non-whitespace character is found. Then it takes as many characters as possible to form a valid integer number representation and converts them to integer value. Anything that is not possible to converts into integer will be discarded/ignored. Thus, we will be looking only at numbers. We start from entry 1 because atoi[0] is reserved for the name of the program.

Next, Ill count grades using count void countGrade(int *entry, int argc) funcrion, which is basically copy and past from problem 3. The only thing that I will add to this function is the following else statement:

```
else if (entry[k] != 0 ||entry[k] != 1 || entry[k] != 2 ||entry[k] != 3 ||entry[k] != 4 || entry[k] != 5)

    printf("Invalid input! Entry is omitted.");
```

```
cout << endl;
```

This will let the user know that invalid number was entered.

When I tested the program I've encountered the problem was that I was using the following loop for(int k=0; k<argc-1; k++) to change enties made by the user into the integers, last entry was omitted all the time. It does not make sense because there are [index-1] elements in the array and that's why I was stopping at index-1. Eventhough it did not make sense, I changed loop to for(int k=0; k<argc; k++) and last entry was counted.

My next problem is that 0 was counted at least once even when zero was not entered. I do not think that I understand whats going on here. However, I was able to fix the problem by changing the int gradesIncrementing[6]={0,0,0,0,0,0} initialization of the array to int gradesIncrementing[6]={-1,0,0,0,0,0};. It fixed the problem, but I was not able to figure out what was going on.