Tatyana Vlaskin
Week 8 Report

PROBLEM 1:

For this problem we need to do the following steps:

1. Create structure called person
2. Give the person structure two members: char name[20] and int age;

```
struct person
{
        char name[20];// member 1
        int age;// member 2
};
```

3. Create a function: person new_person()
4. The function will create an instance of the structure person: person information;
5. The function will also ask the user for the name and age values. This will be accomplished in the following way:

```
cout << "What is your first name? ";
cin.get (information.name, 20);
cout << "How old are you? ";
cin >> information.age;
```

6. Finally there will be a function: integerCheck, that will check if the user made valid entry for age. This is not required for this program, but we've been doing this in the previous assignment, so I've decided to include this function in the program as well. The input will be checked in the following way: integerCheck(information.age).

PROBLEM 2:

For this problem we need to do the following steps:

1. Create structure that will have the member for quiz1, quiz2, midtermExam, finalExam, courseAregage and letterGrade. I'll call my structure: studentRecord:

```
struct studentRecord
{
  int quiz1;
  int quiz2;
  int midterm;
```

```
    int final;
    float average;
    char letterGrade;
};
```

2. Next, we need to create an instance of the structure studentRecord:
   studentRecord grades;
3. The user will be asked to input grades for quiz1, quiz2, midterm and final. Ill make a separate function for this: void getScores();

```
        void getScores(){
        // we need to let the user know that quiz score must be at least 0
        and at most 10.      cout << "Quiz #1 Score: ";
        cin >> grades.quiz1;
        //I'll also have a function that will check the user made a valid entry
        for quiz score
        //the function will check that score is between 0 and 10 inclusive
        Similar thing will be done for quiz2, midterm and final.
        //for the midterm and final we need to let the user know that test
        score must be at least 0 and at most 100.
        There will be a separate function: void testCheck(int &test); that
        will check that the entries made for midterm and final are valid
        }
```

4. Once the scores are entered we need to compute the weighted average. The final exam counts for 50% of the grade, the midterm counts for 25%, and the two quizzes together count for a total of 25%. I also need to normalize quiz score. There will be a separate function: void computeAverage (studentRecord& record);

```
    void computeAverage (studentRecord& record)
    {
     // To normalize the quiz scores,
    //Ill calculate % for each quiz and average % for 2 quizes
       double quiz1Percent, quiz2Percent;
       quiz1Percent = 100 * record.quiz1 / 10.0;
       quiz2Percent = 100 * record.quiz2 / 10.0;
       double quizAverage = (quiz1Percent + quiz2Percent) / 2;
```

```
  // Next step will be to compute the weighted average to get the
numeric course grade
 //I think I'll declare some global constants at this point
const double EXAM = 0.5;
const double MIDTERM= 0.25;
const double QUIZ = 0.25;
//Next step will be to compute course average:
   record.courseAverage = quizAverage * QUIZ + record.midterm *
MIDTERM+   record.final * EXAM;

}
```

5. Next step will be to determine letter grade based on the record.courseAverage. There will be another function for that: char letterGrade (double grade); This function will have a 6 if/else if/else statements:

```
char letterGrade (double grade)
{
 char letter;

 if (grade >= 90)
   letter = 'A';
 else if (grade >= 80)
   letter = 'B';
 else if (grade >= 70)
   letter = 'C';
 else if (grade >= 60)
   letter = 'D';
 else
   letter = 'F';

 return letter;
}
```

6. Finally, there will be another function: void records (studentRecord &record). This function will output the student's record, which consists of two quiz and two exam scores as well as the student's average numeric score for the entire course and final letter grade.

PROBLEM 3:

In this problem we need to create a program that achieves the same goal as problem #2, but we need to check if command line arguments were used to enter the two quiz scores and two exam scores as 4 numerical values in order of quiz, quiz, midterm, final.

1. Ill just take problem # 2 and do some modifications.
2. First of all main() will be changed to : int main(int argc, char* argv[])
3. The program will be able to take in values from the command line and also ask them as user input.
4. If the if(argc == 1), this will be an indication that user decided not to use command line and values are entered as user input
5. Ill make a function: noArguments();, which is copy and paste of the function void getScores() from the problem 2.
6. Also, just for practice, Ill take my function: void computeAverage (studentRecord& record) from problem 2 and change it to lets say function: void normalizeGrades(studentRecord& record)

   First things that Ill do is slightly change my global constant: QUIZ. In problem 2 is was const double QUIZ = 0.25;. So normalize quiz score from 0-10 range to 0-100 range, I have to do the following thing:

   ```
   double quiz1Percent, quiz2Percent;
   quiz1Percent = 100 * record.quiz1 / 10.0;
   quiz2Percent = 100 * record.quiz2 / 10.0;
   double quizAverage = (quiz1Percent + quiz2Percent) / 2;
   ```

   For this problem, Ill change const double QUIZ = 0.25 to const double QUIZ = 1.25; and this will take care of the normalization.
7. Going back to the void normalizeGrades(studentRecord& record) function, it will look something like this:

   ```
   void normalizeGrades(studentRecord& record)
   {
      record.courseAverage =
   (((record.quiz1+record.quiz2)*QUIZ)+(record.midterm*MIDTERM)+
   (record.final*EXAM));

      record.letterGrade = letterGrade (record.courseAverage);
   }
   ```
8. The main method will look like this:

```
if(argc == 1) // this will be an indication that command line is not
used
       {
       noArguments();
       normalizeGrades(grades);
       outputRecord(grades);
       }

       if (argc < 5) { // if command line is used and too few arguments
supplied

      std::cerr << "Not entered all details of student " ;
         return 1;
   }
// if command line is used and enough arguments supplied
   else if(argc==5)
   {
      grades.quiz1=atoi(argv[1]);
      quizCheck(grades.quiz1);// makes sure that quiz score is
between 0-10
      grades.quiz2=atoi(argv[2]);
      quizCheck(grades.quiz2);
      grades.midterm=atoi(argv[3]);
      testCheck(grades.midterm);
      grades.final=atoi(argv[4]);
      testCheck(grades.final);// makes sure that test is between 0-
100
      normalizeGrades(grades);
      outputRecord(grades);
   return 0;
}
// if command line is used and too many arguments supplied
   else if(argc>5)
{
   cout <<" Too many arguments supplied" <<endl;
}
```

I had to remove the following lines from the program because when I tested it on FLIP, it did not let me run the program using the user input method:

```
if (argc < 5) { // if command line is used and too few arguments supplied

    std::cerr << "Not entered all details of student " ;
        return 1;
}
```

PROBLEM 4:

I need to do the following steps:

1. Create a struct variable. Lets say it will be:

```
struct rectangle
{
   int length;
   int width;
};
```

2. Create a pointer to the structure rectangle. Ill initialize it to zero.

```
struct rectangle *pointerRectangle = 0;
pointerRectangle = &rect1;
```

3. Demonstrage that a pointer to the structure works similar to a normal pointer.
   This will be accomplished in the following way:

```
cout<<"The memory address to the struct rectangle is:
"<<pointerRectangle<<endl;
cout<<"Value stored in member variable called length is: "<<pointerRectangle-
>length<<endl;
cout<<"Value stored in member variable called width is: "<<pointerRectangle-
>width<<endl;
cout << "The memory address to stuct rectangle member length is: "
<<&rect1.length <<endl;
cout << "The memory address to stuct rectangle member width is: "
<<&rect1.width <<endl;
```

4. Next step is to create array of the struct (**person friends[5];**), fill in the variables, and access the member variable values.
5. The array will be created like that:

```
person people[5];
```

```
//initialization of the array
   people[0].name = "Katerina";
   people[1].name = "Violeta";
   people[2].name = "Svetlana";
   people[3].name = "Lyubov";
   people[4].name = "Snezhana";
   person *pointerName; // pointer to the members of the array
//to access the members of the array, Ill use a loop:
   for(int i = 0; i < 5; i++)
      {
         pointerName = &people[i];
         cout << pointerName->name << " is member " << i + 1 << " of the
struct array people." << endl;
      }
```

PROBLEM 5:

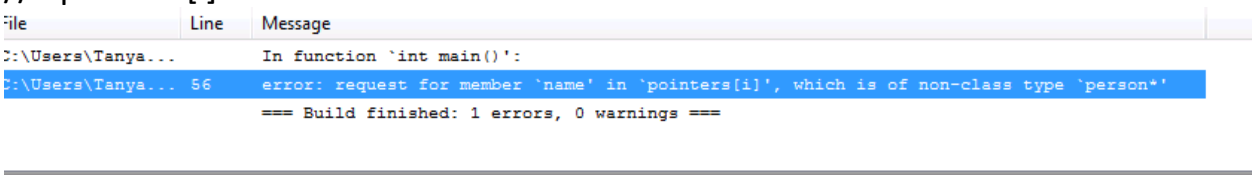
We need to do the following things for this problem:

1. Create an array of pointers:
   ```
   person *pointers[5];
   pointers[i]=&people[i];
   ```
2. Fill in the array with names:
   ```
   person people[5];
   people[0].name = "Katerina";
   people[1].name = "Violeta";
   people[2].name = "Svetlana";
   people[3].name = "Lyubov";
   people[4].name = "Snezhana";
   ```
3. Display the pointer to the memebers as well as access the member values

According to the description of the problem we should be able to access the members of the structure using the arrow notation (->) or by dereferencing with the star (*) and then using a dot (.). So I tried to use both ways to access the members and only arrow notation has worked.

```
for(int i = 0; i < 5; i++)
{
   pointers[i]=&people[i];
cout<<"Pointer at index "<<i <<" of the array of pointers points to memory
address: "<<pointers[i] <<endl;
```

```
cout <<"The value/member stored in that address is: "<<pointers[i]->name
<<endl <<endl;
// *pointers[i].name GAVE ME AN ERROR:
```




```
cout <<"The value/member stored in that address is: "<<*pointers[i].name
<<endl <<endl;

   }// THIS WAS REMOVED FROM THE PROGRAM
```

I was not able to figure out what was going on and why it does not work, so I removed that line from my code because the problem does not require us to access members of the structure using both ways. However, I really want to know why it did not work. I'll post this question on the discussion board.

PROBLEM 6:

For this problem we have choose any program and make it command line arguments friendly. I already made program 3 to be command line argument friendly. However, I'll also make, lets say problem 3 to be command line argument friendly. The user will be able to type numbers, strings, words, in the command line. All the entries will be stored as a string type. Once the user is done with entries and presses enter, the following information will be displayed to the user:

```
 cout<<"Pointer at index "<<i <<" of the array of pointers points to memory
address: "<<pointers[i]          cout <<"The entered name stored in that address is:
"<<pointers[i]->name
```

The program will look like that:

```
int main(int argc, char* argv[])
{
   if (argc ==1 ) {  // THIS WILL BE COPPY AND PAST FROM PROBLEM 3
      cerr << "Not entered all details to run the program " <<endl;
      return 1;
   }
   else// IF AT LEAST ONE ARGUMENT WAS ENTERED
```

```
  {
     person * people; // creates dynamic array to enter first names. I DECIDED TO
USE DYNAMIC ARRAYS
//BECAUSE I DO NOT KNOW HOW MANY NAMES/NUMBERS/ETC THE USER WILL
ENTER
     people = new person [argc];
     // fills in array with the names that are entered by the user
     for(int index = 1; index<argc; index++)
     {
        people[index].name = (argv[index]);
     }

     person *pointers[argc]; // Allocates for an array of argc pointers

     for(int i=1;i<argc;i++) // please note that we start at 1 because 0 is reserved
for name of the program
     {
        pointers[i]=&people[i];
       MESSAGE TO THE USER
     }
```