

CSCI 140 -- Lab Final (Revised Version)

Name: Ali Mortada

Make sure to read all instructions before attempting this assignment. You cannot work with another student or communicate with anyone for this assignment. If you used an online solution, a solution on site such as Discord, or someone else solution, 0 will be given. If you posted/shared a solution and someone else uses it, you might get a 0 as well. You will only need to submit 2 out of 3 problems on lab final day.

Each problem is worth 20 points and it is best to work on all 3 problems first. You will need to modify and submit 2 revised problems based on the new requirements on lab final day.

Revised instructions:

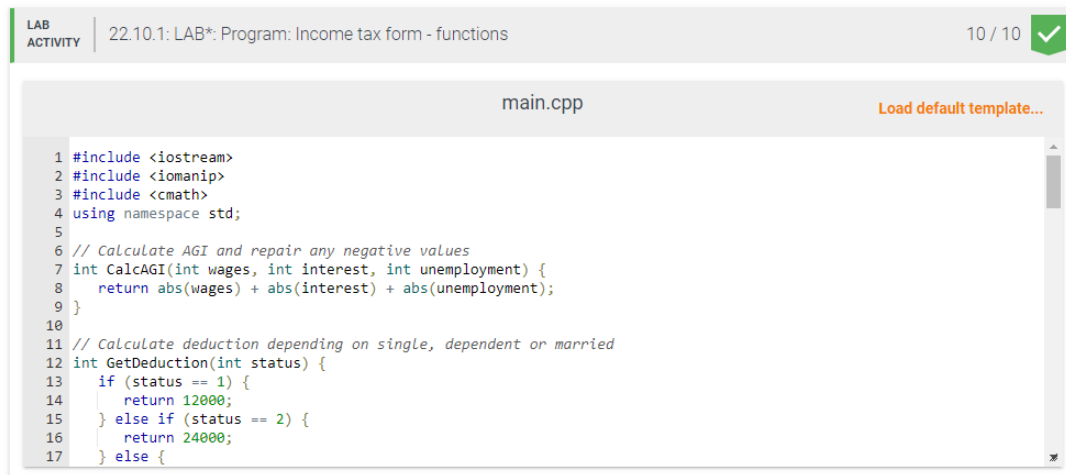
You must use the provided C++ template from Canvas and output "Author: Your Name(s)" for all new programs. If you are modifying an existing program, use "// Modified by: Your Name(s)".

Select 2 problems to modify based on the new requirements and you only need to submit required information as specified below for each problem. It is your responsibility to confirm that you submitted your file correctly, so it is best to view your submission to make sure it was submitted correctly. You must submit by the deadline or 0 will be given for this assignment.

Problems modified: Exercise 1 and Exercise 2

Exercise 1 – 22.10 LAB*: Program: Income tax form – functions

Copy/paste a screenshot showing the current score below:



```
1 #include <iostream>
2 #include <iomanip>
3 #include <cmath>
4 using namespace std;
5
6 // Calculate AGI and repair any negative values
7 int CalcAGI(int wages, int interest, int unemployment) {
8     return abs(wages) + abs(interest) + abs(unemployment);
9 }
10
11 // Calculate deduction depending on single, dependent or married
12 int GetDeduction(int status) {
13     if (status == 1) {
14         return 12000;
15     } else if (status == 2) {
16         return 24000;
17     } else {
```

Revised instructions: Copy/paste the source code to your development environment. Modify the program to support the new requirements below:

Income	Tax for Dependent or Single Filers
\$0 - \$10000	10% of the income
\$10001 - \$40000	\$1000 + 12% of the amount over \$10000
\$40001 - \$85000	\$4600 + 22% of the amount over \$40000
\$85001 - \$15000	\$14500 + 24% of the amount over \$85000
over \$150000	\$30100 + 30% of the amount over \$150000
Income	Tax for Married Filers
\$0 - \$20000	10% of the income
\$20001 - \$80000	\$2000 + 12% of the amount over \$20000
\$80001 - \$160000	\$9200 + 22% of the amount over \$80000
over \$160000	\$26800 + 27% of the amount over \$160000

Run the following test cases and you should confirm that they work correctly.

```
100000 0 0 1 20000
200000 0 0 1 40000
```

```
100000 0 0 2 15000
200000 0 0 2 35000
```

Copy/paste source code and a screenshot showing the output below:

```
// Zybooks 22.10 LAB*: Program: Income tax form -- functions
// Modified by: Ali Mortada

#include <iostream>
#include <iomanip>
#include <cmath>
using namespace std;

// Calculate AGI and repair any negative values
int CalcAGI(int wages, int interest, int unemployment) {
    return abs(wages) + abs(interest) + abs(unemployment);
}

// Calculate deduction depending on single, dependent or married
int GetDeduction(int status) {
    if (status == 1) {
        return 12000;
    } else if (status == 2) {
        return 24000;
    } else {
        return 6000;
    }
}

// Calculate taxable but not allow negative results
int CalcTaxable(int agi, int deduction) {
    int taxable = agi - deduction;
    if (taxable < 0) {
        return 0;
    }
    return taxable;
}

// Calculate tax for single or dependent
int CalcTax(int status, int taxable) {
    double tax = 0.0;
    if (status == 2) { // Case for married filers first
        if (taxable >= 0 && taxable <= 20000) {
            tax = taxable * .10;
        } else if (taxable >= 20001 && taxable <= 80000) {
            tax = 2000 + ((taxable - 20000) * .12);
        }
    }
}
```

```

    } else if (taxable >= 80001 && taxable <= 160000) {
        tax = 9200 + ((taxable - 80000) * .22);
    } else {
        tax = 26800 + ((taxable - 160000) * .27);
    }
} else {
    // Case for dependent/single filers second
    if (taxable >= 0 && taxable <= 10000) {
        tax = taxable * .10;
    } else if (taxable >= 10001 && taxable <= 40000) {
        tax = 1000 + ((taxable - 10000) * .12);
    } else if (taxable >= 40001 && taxable <= 85000) {
        tax = 4600 + ((taxable - 40000) * .22);
    } else if (taxable >= 85001 && taxable <= 150000) {
        tax = 14500 + ((taxable - 85000) * .24);
    } else {
        tax = 30100 + ((taxable - 150000) * .30);
    }
}

tax = round(tax);

return static_cast<int>(tax);
}

// Calculate tax due and check for negative withheld
int CalcTaxDue(int tax, int withheld) {
    if (withheld < 0)
        withheld = 0;

    return tax - withheld;
}

int main() {
    int wages, interest, unemployment, status, withheld;
    int tax, agi, due, deduction, taxable;

    // Step #1: Input information
    cin >> wages >> interest >> unemployment >> status >> withheld;

    // Step #2: Calculate AGI
    agi = CalcAGI(wages, interest, unemployment);
    cout << "AGI: $" << agi << endl;

    // Step #3: Get deduction AGI
    deduction = GetDeduction(status);

```

```

    cout << "Deduction: $" << deduction << endl;

    // Step #4: Calculate taxable income
    taxable = CalcTaxable(agi, deduction);
    cout << "Taxable income: $" << taxable << endl;

    // Step #5: Calculate federal tax
    tax = CalcTax(status, taxable);
    cout << "Federal tax: $" << tax << endl;

    // Step #6: Calculate tax due
    due = CalcTaxDue(tax, withheld);
    cout << "Tax due: $" << due << endl;

    return 0;
}

```

```

100000 0 0 1 20000
AGI: $100000
Deduction: $12000
Taxable income: $88000
Federal tax: $15220
Tax due: $-4780

```

```

200000 0 0 1 40000
AGI: $200000
Deduction: $12000
Taxable income: $188000
Federal tax: $41500
Tax due: $1500

```

```

100000 0 0 2 15000
AGI: $100000
Deduction: $24000
Taxable income: $76000
Federal tax: $8720
Tax due: $-6280

```

```

200000 0 0 2 35000
AGI: $200000
Deduction: $24000
Taxable income: $176000
Federal tax: $31120
Tax due: $-3880

```

Exercise 2 – 23.18 LAB: Find student

Copy/paste source code and a screenshot showing the score for 23.17 below:



The screenshot shows a code editor window titled "23.17.1: LAB: Find student with highest GPA" with a progress indicator "10 / 10" and a green checkmark. The current file is "Course.cpp". The code is as follows:

```
1 #include <iostream>
2 #include "Course.h"
3 using namespace std;
4
5 Student Course::FindStudentHighestGpa() {
6     Student highestStudent = roster[0];
7     double highestGPA = highestStudent.GetGPA();
8
9     for (int i = 0; i < roster.size(); i++) {
10         if (roster[i].GetGPA() > highestGPA) {
11             highestGPA = roster[i].GetGPA();
12             highestStudent = roster[i];
13         }
14     }
15     return highestStudent;
16 }
17
```

```
// Zybooks 23.17 LAB: Find student with highest GPA
// Modified by: Ali Mortada
```

```
#include <iostream>
#include "Course.h"
using namespace std;

Student Course::FindStudentHighestGpa() {
    Student highestStudent = roster[0];
    double highestGPA = highestStudent.GetGPA();

    for (int i = 0; i < roster.size(); i++) {
        if (roster[i].GetGPA() > highestGPA) {
            highestGPA = roster[i].GetGPA();
            highestStudent = roster[i];
        }
    }
    return highestStudent;
}

void Course::AddStudent(Student s) {
    roster.push_back(s);
}
```

Revised instructions: Work on 23.18. It is the same as 23.17 which has an additional requirement. Need to add a member function `DisplayStudentByLastName()` to `Student` class.

Copy/paste source code and a screenshot showing the score below:



The screenshot shows a C++ IDE interface. At the top, there is a header bar with 'LAB ACTIVITY' on the left, '23.18.1: LAB: Find student' in the center, and '10 / 10' with a green checkmark icon on the right. Below the header, the current file is 'Course.cpp'. The main area displays the following C++ code:

```
15     return highestStudent;
16 }
17
18 void Course::AddStudent(Student s) {
19     roster.push_back(s);
20 }
21
22 void Course::DisplayStudentByLastName(string lastName) {
23     int count = 0;
24     for (int i = 0; i < roster.size(); i++) {
25         if (roster[i].GetLast() == lastName) {
26             cout << roster[i].GetFirst() << " " << roster[i].GetLast() << " (" << roster[i].GetGPA() << ")" << endl;
27             count++;
28         }
29     }
30     cout << "Found " << count << " students." << endl;
31 }
```

```

// Zybooks 23.18 LAB: Find student
// Modified by: Ali Mortada

#include <iostream>
#include "Course.h"
using namespace std;

Student Course::FindStudentHighestGpa() {
    Student highestStudent = roster[0];
    double highestGPA = highestStudent.GetGPA();

    for (int i = 0; i < roster.size(); i++) {
        if (roster[i].GetGPA() > highestGPA) {
            highestGPA = roster[i].GetGPA();
            highestStudent = roster[i];
        }
    }
    return highestStudent;
}

void Course::AddStudent(Student s) {
    roster.push_back(s);
}

void Course::DisplayStudentByLastName(string lastName) {
    int count = 0;
    for (int i = 0; i < roster.size(); i++) {
        if (roster[i].GetLast() == lastName) {
            cout << roster[i].GetFirst() << " " << roster[i].GetLast() << " (" <<
roster[i].GetGPA() << ")" << endl;
            count++;
        }
    }
    cout << "Found " << count << " students." << endl;
}

```


Exercise 3 – Large Integers version 3

Modify your Large Integers version 2 (PA 5) to add a large integer with a single digit integer (0 to 9). Set up the following function to add a large integer with a single digit integer and use it:

```
void addDigit(const int opl[], int digit, int result[])
```

Follow the interface below and you must try the following test cases (second operand is an int value so can be stored in an int variable):

```
Enter an expression --> 1235 + 7<Enter>
1235 + 7 = 1242
```

```
Enter an expression --> 987654321 + 0<Enter>
987654321 + 0 = 987654321
```

```
Enter an expression --> 9999999999999999999 + 1<Enter>
9999999999999999999 + 1 = 10000000000000000000
```

Revised instructions: Modify your code to add another function to subtract a large integer by a single digit integer (0 to 9). You can assume that a large integer has at least one digit and will be at least as large as the one digit. Set up the following functions to subtract a large integer by a single digit integer and use it:

```
void subDigit(const int opl[], int digit, int result[]) // can also use vectors
```

Try the following test cases:

Author: [Your Name]

```
Enter an expression --> 1235 + 7<Enter>
1235 + 7 = 1242
```

```
Enter an expression --> 987654321 + 0<Enter>
987654321 + 0 = 987654321
```

```
Enter an expression --> 9999999999999999999 + 1<Enter>
9999999999999999999 + 1 = 10000000000000000000
```

```
Enter an expression --> 1235 - 9<Enter>
1235 - 9 = 1226
```

```
Enter an expression --> 5 - 5<Enter>
5 - 5 = 0
```

```
Enter an expression --> 10000000000000000000 - 1<Enter>
10000000000000000000 - 1 = 9999999999999999
```

Copy/paste source code and input/output below: