IF184101 Basic Programming (IUP)

Final Exam

Starting date:	20 November 2020
Deadline:	27 November 2020, 23:59 WIB. **Penalty: 0.15% of grade/minute of tardiness.**
Exam type:	Open
Send to:	MM Irfan Subakti <yifana@gmail.com> CC to Rosa Valentine Lammora <rosalammora@gmail.com>, Dicksen Alfersius Novian <dicksenan@gmail.com>, James Rafferty Lee <jamesrafe10@gmail.com>, Michael Ricky <michaelricky007@gmail.com>, Clement Prolifel Priyatama <clement.085w@gmail.com> with the subject: IF184101_BASPRO_IUP_FIN_StudentID_StudentName
File type and format:	A zip file containing all of the .c & .cpp source files & the declaration
Filename format:	IF184101_BASPRO_IUP_FIN_StudentID_StudentName.ZIP

Instruction
Please do these steps as in the following.

1. Create a program to solve the following problem and name it `01_magic_herb.c`. [**25 points**]]

   The boy has safely returned to his home with his hard-worked spoils, a magical herb. That herb is said to be the key to cure his sickly mother. He was happy, he already imagined the face of his energetic mother, but he doesn't know how to concoct the medicine from the herb. He only read it from a book, that his father left, that the herb was the key ingredient.

   He was confused, it's so close yet so far. He reread his books, again and again, to check if he misses something, without sleep nor eat for straight three days and at long last, he found a gibberish number that hidden between the pages. That's the magic word to activate the true nature of the book! The boy yells happily reminiscing the figure of his long-lost father. As if remembering something the boy went to his father study and took a piece of paper.

   The number is
   `{94,222,221,186,181,208,207,110,187,185,17,212,115,215,100}`

   The method to decipher the gibberish number that is written in that paper

   You must proceed the number one by one from the rightmost number, i.e., 100, to the leftmost one, i.e., 94. It means the deciphering procedure is started from the right to the left. The rightmost number, 100, is the first number (n_0) to decipher the gibberish number. By considering the next number (remember, we move from the right to the left), i.e., 215, then we are going to find the **designated number (n_i) to be deciphered**. Finding

that **designated number** can be done by performing one of the four operations, as in the following.

Operation 1:

$$n_{i+1} = n_{i+1} + n_i$$

Operation 2:

$$n_{i+1} = n_{i+1} - n_i$$

Operation 3:

$$n_{i+1} = (n_{i+1} * 2) - n_i$$

Operation 4:

$$n_{i+1} = (n_{i+1} - 3) * 2 + n_i$$

Which from those operations the **designated number** is whether n_{i+1} is between 65 (inclusive) and 122 (inclusive) or not? In other words, whether $65 \leq n_{i+1} \leq 122$ (between 'A' and 'z') or not?

Example. Remember, the deciphering process starts from the right to the left.

1st iteration

i = 0 → n_0 = 100, n_1 = 215

Operation 1:

n_1 = 215 + 100 = 315 (false, it's not the **designated number**)

Operation 2:

n_1 = 215 - 100 = **115** (true, the number is between 65 – 122, so **115** is the **designated number**. The ASCII code for **115** is **s**, so **s** is the last **deciphered character** of the magic word)

Operation 3:

n_1 = (215 * 2) - 100 = 330 (false, it's not the **designated number**)

Operation 4:

n_1 = (215 – 3) * 2 + 100 = 524 (false, it's not the **designated number**)

We got the **designated number**, i.e., **115**, and the current number now is below.

`{94,222,221,186,181,208,207,110,187,185,17,212,115,115,100}`

2nd iteration

i = 1 → n_1 = 115, n_2 = 115

Operation 1:

n_2 = 115 + 115 = 230 (false, it's not the **designated number**)

Operation 2:

n_2 = 115 - 115 = 0 (false, it's not the **designated number**)

Operation 3:

n_2 = (115 * 2) - 115 = 115 (true, the number is between 65 – 122, so **115** is the **designated number**. The ASCII code for **115** is **s**, so **s** is the second last **deciphered character** of the magic word)

Operation 4:

n_2 = (115 – 3) * 2 + 115 = 339 (false, it's not the **designated number**)

We got the **designated number**, i.e., **115**, and the current number now is below.

{94,222,221,186,181,208,207,110,187,185,17,212,**115,115,100**}

3rd iteration

i = 2 → n_2 = 115, n_3 = 212

Operation 1:

n_3 = 212 + 115 = 327 (false, it's not the **designated number**)

Operation 2:

n_3 = 212 - 115 = **97** (true, the number is between 65 – 122, so **97** is the **designated number**. The ASCII code for **97** is **a**, so **a** is the third last **deciphered character** of the magic word)

Operation 3:

n_3 = (212 * 2) - 115 = 309 (false, it's not the **designated number**)

Operation 4:

n_3 = (212 – 3) * 2 + 115 = 533 (false, it's not the **designated number**)

We got the **designated number**, i.e., **97**, and the current number now is below.

{94,222,221,186,181,208,207,110,187,185,17,**97,115,115,100**}

…… and so, on

Solve it by using the recursive function. Note: the last number (the rightmost one, n_0 = 100) produced the last **deciphered character 's'** of the **designated number '115'** from the result of the operation 2 in the 1st iteration. It means we got the magic word " … s". Then by the similar procedure, we got the magic word " … ass" after 2nd iteration (n_1 = 215 by operation 3) and 3rd iteration (n_2 = 115 by operation 2), respectively.

Output

Print the ASCII character of the **designated number**. Hint: you can use `printf("%c\n", number)` to print the ASCII character of the designated number.

Example output:

```
?
?
?
.
.
.
.
.
.
.
.
.
a
s
s
```

2. Please create a program, namely `02_chess_pieces.cpp`. At the beginning of the program, please write down the codes as in the following. Then continue the codes for `class Rook`, `Bishop`, `Queen`, and `Knight` like the `Pawn` class below. All classes should have `x = 0`, `y = 0` as their initial locations. [**25 points**]

   Code:

```
#include <iostream>
#include <cmath>

using namespace std;

class Piece {
protected:
    int faction; //faction is 1 for white and -1 for black
    int x; int y;
    char * name;
public:
    void getLocation(){
        cout << name << " is at x: " << x << " y: " << y << endl;
    }
    bool moveCondition (int x_dest, int y_dest);
    void move(int x_dest, int y_dest;
};
class Pawn : public Piece {
public:
    Pawn (int f) { //constructor is not inherited
        name = "Pawn";
        faction = f;
        x = 0;
        y = 0;
    }
    bool moveCondition(int x_dest, int y_dest) {
        return ((x_dest == x) && ((y_dest - y)==(1 * faction)));
    }
    void move(int x_dest, int y_dest) {
        if (moveCondition (x_dest, y_dest)) {
            x = x_dest;
            y = y_dest;
            getLocation();
        } else {
            cout << "invalid move" << endl;
        }
    }
};
```

   Hint:

   - Move condition for `Rook` is in the following.
     `((x_dest != x) ^ (y_dest != y))`
   - Move condition for `Bishop` is in the following.
     `(((x_dest - x) == (y_dest - y)) && (y_dest - y != 0) && (x_dest - x != 0))`
   - Move condition for `Knight` is in the following.

```
((abs(x_dest - x) == 2) && (abs(y_dest - y) == 1 ) ||
(abs(x_dest - x) == 1) && (abs(y_dest - y) == 2 ))
```

- Move condition for `Queen` is in the following.

```
((abs(x_dest - x) == abs(y_dest - y)) && ((y_dest -
y != 0) && (x_dest - x != 0)) ||  ((x_dest != x) ^
(y_dest != y)))
```

3. Please create a program, namely `03_overloading.cpp`. At the beginning of the program, please write down the codes as in the following. Then continue the codes by creating overloading of the class for adding three, four, and five variables. [**25 points**]
   Hint:

```
int add(int a, int b){
      return a+b;
}
```

4. Please create a program, namely `04_anagram.cpp`. The main purpose of the program is to search the total number of the pattern and its anagram inside a string and print them alongside their location. Please utilize a function called is_permutation() in a C++. [**25 points**]

   Example:

   Input:

```
String: aabaabaa
Pattern: aaba
```

   Output:

```
Anagram:
aaba at index 0
abaa at index 1
aaba at index 3
abaa at index 4
```

   Hint:

   Please use the almighty header `<bits/stdc++.h>`

5. To avoid plagiarism/cheating, every student needs to pledge and declare, then she/he must submit her/his **signed pledge and declaration** as in the following. Failed to do so will be resulted in getting 0 (zero) grade. Attach the **scanned**/**photo** of your *declaration* in your report.

   "By the name of Allah (God) Almighty, herewith I pledge and truly declare that I have solved final exam by myself, didn't do any cheating by any means, didn't do any plagiarism, and didn't accept anybody's help by any means. I am going to accept all of the consequences by any means if it has proven that I have been done any cheating and/or plagiarism."

   [Place, e.g., Surabaya], [date, e.g., 27 November 2020]

   <Signed>

   [Full name, e.g., Ayuna Dewi Aleksandra]
   [StudentID, e.g., 502520xxxx]

6. ZIP the files of `01_magic_herb.c`, `02_chess_pieces.cpp`, `03_overloading.cpp`, `04_anagram.cpp` and your declaration (e.g., `Declaration.PDF`) into 1 (one) only .ZIP file, namely `IF184101_BASPRO_IUP_FIN_StudentID_StudentName.ZIP`. Send this .ZIP file to `yifana@gmail.com` and CC-ed to `rosalammora@gmail.com`, `dicksenan@gmail.com`, `jamesrafe10@gmail.com`, `michaelricky007@gmail.com`, `clement.085w@gmail.com`.

7. Have a great lovely, wonderful day, guys! Good luck! 😊