1THE UNIVERSITY OF HONG KONG

COMP1117 Computer Programming I

Assignment 3

Deadline: 5:00pm, Nov 4, 2017

You can assume that the input of your program is valid, and your programs do not need to check the validity of the input.

1. [50%] This question asks you to implement three functions on 2D arrays. Suppose in your program you have defined the constant variable SIZE as follows:

```
const int SIZE = 4;
```

Note that when the TAs compile your program, they may change the value 4 to some other value.

Define the following three functions for the three common operations for arrays, namely add, multi and transpose:

a) void transpose(int A[SIZE][SIZE], int B[SIZE][SIZE]) ;
// find the transpose of A.
Function Task: Flip the content of A over its diagonal, which covers the entries A[i][j] for $0 \leq i \leq SIZE - 1$. For example, suppose the content of A is

```
1 2 3 4
5 6 7 8
0 1 2 3
4 5 6 7
```

then when the function returns, the content of B is

```
1 5 0 4
2 6 1 5
3 7 2 6
4 8 3 7
```

In general, B[i][j] = A[j][i]
for $0 \leq i \leq SIZE - 1$, and $0 \leq j \leq SIZE - 1$;

b) void add(int A[SIZE][SIZE], B[SIZE][SIZE], C[SIZE][SIZE]) ;
// find the addition of A and B.
Function Task: add two arrays A and B such that when the function returns
C[i][j] = A[i][j] + B[i][j];
for $0 \leq i \leq SIZE - 1$, and $0 \leq j \leq SIZE - 1$;

c) void multi(int A[SIZE][SIZE], B[SIZE][SIZE], C[SIZE][SIZE]);
// find the multiplication and A and B.
Function Task: When the function returns, we have

$$C[i][j] = \sum_{k=0}^{SIZE-1} A[i][k] \times B[k][j]$$

for $0 \le i \le SIZE - 1$, and $0 \le j \le SIZE - 1$;

Write a program myarray.cpp that implements the three functions for the three array operations transpose, addition and multiplication, and then repeatedly asks the user to input an integer from 0 to 3, and then call the corresponding function to perform the operation until a 0 is read. More specifically,

- If the input is 0, stop and exit the program.
- If the input is 1, then read a 2D array of dimension SIZE $\times$ SIZE, and then output the transpose of the array.
- If the input is 2, then read two 2D arrays of dimension SIZE $\times$ SIZE, and then outputs the sum of the two arrays.
- If the input is 3, then read two 2D arrays of dimension SIZE $\times$ SIZE, and then outputs the product of the two arrays.

Sample runs:
Suppose that SIZE is set to 4.

```
Please input the number of an operation (0.Stop; 1.Transpose; 2.Addition; 3.Multiplication):
1
Please input an 4*4 two-dimensional array of integer:
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 16
Output:
1 5 9 13
2 6 10 14
3 7 11 15
4 8 12 16
Please input the number of an operation (0.Stop; 1.Transpose; 2.Addition; 3.Multiplication):
2
Please input two 4*4 two-dimensional arrays of integer:
1 1 1 1
1 2 3 4
4 3 2 1
2 2 2 2
4 5 6 7
3 2 1 2
5 3 2 0
9 7 6 5
Output:
5 6 7 8
4 4 4 6
9 6 4 1
11 9 8 7
Please input the number of an operation (0.Stop; 1.Transpose; 2.Addition; 3.Multiplication):
3
Please input two 4*4 two-dimensional arrays of integer:
1 1 1 1
1 2 3 4
4 3 2 1
2 2 2 2
4 5 6 7
3 2 1 2
5 3 2 0
9 7 6 5
Output:
21 17 15 14
61 46 38 31
44 39 37 39
42 34 30 28
Please input the number of an operation (0.Stop; 1.Transpose; 2.Addition; 3.Multiplication):
0
```

Input/Output format:
<"Please input the number of an operation (0.Stop; 1.Transpose; 2.Addition;
3.Multiplication):"><newline>
< an operator><newline>
If <an operator> is 1:
<A_{00} ><space><A_{01}><space><A_{02} ><space><A_{03}><newline>
<A_{10} ><space><A_{11}><space><A_{12} ><space><A_{13}><newline>
<A_{20} ><space><A_{21}><space><A_{22} ><space><A_{23}><newline>
<A_{30} ><space><A_{31}><space><A_{32} ><space><A_{33}><newline>
If <an operator> is 2 or 3:
<A_{00} ><space><A_{01}><space><A_{02} ><space><A_{03}><newline>
<A_{10} ><space><A_{11}><space><A_{12} ><space><A_{13}><newline>
<A_{20} ><space><A_{21}><space><A_{22} ><space><A_{23}><newline>
<A_{30} ><space><A_{31}><space><A_{32} ><space><A_{33}><newline>
<B_{00} ><space><B_{01}><space><B_{02} ><space><B_{03}><newline>
<B_{10} ><space><B_{11}><space><B_{12} ><space><B_{13}><newline>
<B_{20} ><space><B_{21}><space><B_{22} ><space><B_{23}><newline>
<B_{30} ><space><B_{31}><space><B_{32} ><space><B_{33}><newline>
If <an operator> is 0: return
<"Output:"><newline>
<C_{00} ><space><C_{01}><space><C_{02} ><space><C_{03}><newline>
<C_{10} ><space><C_{11}><space><C_{12} ><space><C_{13}><newline>
<C_{20} ><space><C_{21}><space><C_{22} ><space><C_{23}><newline>
<C_{30} ><space><C_{31}><space><C_{32} ><space><C_{33}><newline>

2. [50%] We say that **a positive integer m is interesting for a non-zero digit d** (i.e., d = 1, 2, 3, 4, 5, 6, 7, 8, 9) if either (i) m is divisible by d or (ii) at least one digit of m is equal to d. Write a program **interesting.cpp** to read an integer $N \geq 0$, and then repeat the following task N times:

Read a positive integer m and a non-zero digit d, then output "interesting" if m is interesting for d, and output "not interesting" otherwise.

Note that:

I. Your program cannot use array or string.

II. Your program should include the following function:
bool judgeDivisible (unsigned int inter, unsigned int digit) {
// It returns whether the inter can be divisible by the digit.
// For example, if inter is 13 and digit is 3, then the function returns False.
}

III. Your program should include the following function:
bool judgeDigitEqual (unsigned int inter, unsigned int digit) {
// It returns whether at least one digit of inter is equal to the digit.
// For example, if inter is 13 and digit is 3, then the function returns True.
}

IV. Your program should include the following function:
bool judgeInteresting (unsigned int inter, unsigned int digit) {

// It returns whether the inter is interesting for the digit.
// For example, if inter is 13 and digit is 3, then the function returns True.
}

Sample runs:

```
3
13 3
interesting
23123 4
not interesting
14 7
interesting
```

Input/Output format:

<The number of repetitions N> <enter>
<$inter_0$> <space> <d_0> <newline>
For i^{th} line output,
<"interesting/on interesting"> <new line>
< $inter_1$> <space><d_1><newline>
For i^{th} line output,
<"interesting/on interesting"> <new line>
…
< $inter_{N-1}$><space><d_{N-1}><newline>
<"interesting/on interesting"> <new line>

Important Notes:

i. You must write your name and university number as part of the comments at the beginning of your program source code.
ii. Your programs must follow all the requirements mentioned in the specification.
iii. Your program must follow the formats of the sample input / output strictly.
iv. Your program should be properly indented and documented.
v. You should only submit source codes (*.cpp), not executables.
vi. You must use the program names as suggested in the specification, i.e., myarray.cpp and interesting.cpp.
vii. Please make sure that your source code could be compiled in Code::Blocks environment (with the settings mentioned in workshop) before submission.

Hand in:

Submit your programs (myarray.cpp and interesting.cpp) electronically using the Moodle system to Assignment 3 – myarray.cpp and Assignment 3 – interesting.cpp correspondingly under Assignments section. Late submission will NOT be accepted.