Programming Design, Spring 2014

Suggested Solution for Homework 03

Solution provider: Christine Hsieh

Problem 1 (10 points)

(a) (2 points)

The output is the *n*-th term of the Fibonacci series. For example, when *n* equal to 10, the output will be 55, which is the 10th term of the Fibonacci series.

此程式會輸出費氏數列的第n項。以題目n=10為例，程式會輸出55，也就是費氏數列的第10項。

(b) (3 points)

In this case, reduce one of the assignment in the loop is a way to make the program run faster. You can see the differences as follow.

The original version.

```
int n = 10;
int num1 = 1;
int num2 = 1;

for (int i = 2; i < n; i++){
    num1 = num1 + num2;
    int temp = num2;
    num2 = num1;
    num1 = temp;
}
cout << num2;
```

The faster version.

```
int n = 10;
int num1 = 1;
int num2 = 1;

for (int i=2; i<n; i++){
    int temp = num1 + num2;
    num2 = num1;
    num1 = temp;
}
cout << num2;
```

(c) (3 points)

```
int n = 10;

//output the n-th term of the Fibonacci series
//by using an array
int arr[10] = {1,1};

for (int i = 2; i < n; i++)
    arr[i] = arr[i-2] + arr[i-1];

cout << arr[n-1];
```

(d) (2 points)

It is because that Fibonacci series increase dramatically while n increases. As n reach 50, the output will be greater than the maximum value that can be stored in an integer variable, which causes the problem called "integer overflow" that makes the output with n = 50 negative.

因為 F_{50}= 12586269025 > 2147483647（$2^{31}-1$）超過了int32可儲存的最大值，造成"整數溢位"(integer overflow)的現象，使得輸出值變為負數。

Problem 2 (10 points)

(a) (5 points)

There are n^3 multiplications that must be performed in such a function of n.

(b) (5 points)

Algorithm for diagonal matrices multiplication
Given two n×n diagonal matrices A and B
Declare an n×n matrix C and initialize all its elements to 0
for i from 1 to n
Update C_{ii} to $A_{ii}B_{ii}$
end for

As the algorithm shows, there are n multiplications that must be performed.

(c) (10 bonus points)

Algorithm for upper triangular matrices multiplication
Given two n×n upper triangular matrices A and B
Declare an n×n matrix C and initialize all its elements to 0
for i from 1 to n
for j from i to n
for k from i to j
Update C_{ij} to $C_{ij}+A_{ik}B_{kj}$
end for

As the algorithm shows, the times of multiplications that must be performed would be as follow:

$$1\times n+2\times(n-1)+\cdots+(n-1)\times 2+n\times 1=\frac{n(n+1)(n+2)}{6}$$

Problem 3 (10 bonus points)

If you execute the code, you might get a similar result as follow.

```
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
2293576 1983684148 2293624 4198887 1 5705520 5707368 -1 2293616 1983649184
4194304 5707368 0 2147311616 2293640 4198968 1 0 2293652 1975925829
2147311616 2293716 1999190005 2147311616 1895842743 0 0 2147311616 0 0
0 2293664 0 -1 1998905581 101363227 0 2293740 1999189960 4198944
2147311616 0 0 0 0 4198944 2147311616 0 2020893505 32
1 12320 220 0 32 0 20 1 7 52
380 1 0 0 0 0 0 2 438759246 664
68 736 608 0 -214797894 1344 74 1420 798 0
760826203 2220 50 2272 768 0 852421325 3040 66 3108
822 0 944791496 3932 94 4028 872 0 -1322777276 4900
86 4988 920 16 9 236 2 1 380 5584
1 2 5964 1204 1 3 7168 3724 1 4
10892 788 2 5 11680 152 2 6 11832 204
2 7 12036 240 1 9 12276 40 2 11
12316 4 1 1682469715 44 1 1 1 6 140
1 5528 44 94 94 0 0 0 0 0
0 0 0 0 2 36 56 0 3801155 5701724
7209065 7274596 7536759 5701724 7209065 7864403 6029427 0 438759246 284
68 356 608 1 -214797894 964 74 1040 798 2
760826203 1840 50 1892 768 3 852421325 2660 66 2728
822 4 944791496 3552 94 3648 872 5 -1322777276 4520
86 4608 920 6 6881357 7471203 7536751 6684783 3014772 6881367
6553710 7798895 3014771 7929939 7602291 7143525 7274563 7340141 7602273 6422633
6619244 0 108 1 268 464 2 44 732 -1588790361
30016548 1 0 0 0 0 0 0 1 0
0 0 184 778 0 0 0 0 0 6881357
7471203 7536751 6684783 3014772 6881367 6553710 7798895 3014771 7929939 7602291
7143525 7274563 7340141 7602273 6422633 6619244 7340076 7274610 6619235 7536755
7471215 7471169 6815843 7602281 6488165 7667828 6619250 2228285 3670136 2228278
7340076 6422645 6881388 4915299 7929957 7274580 6619243 3997806 3538978 3735605
```

The reason why we get such a result is that C++ compilers generally do not generate code to check array bounds, for the sake of efficiency. Out-of-bounds array accesses result in "undefined behavior", and one possible outcome is that "it works". It's not guaranteed to cause a crash or other diagnostic, but if you're on an operating system with virtual memory support, and your array index points to a virtual memory location that hasn't yet been mapped to physical memory, your program is more likely to crash.

Problem 4 (10 points)

See the file “PD14-03.cpp”.