Thomas Nguyen
COEN 171
Professor Vu

Assignment #8

Problem Sets

9.2 In most Fortran IV implementations, all parameters were passed by reference, using access path transmission only. State both the advantages and disadvantages of this design choice.

Advantage:

- If passed by reference, subprograms have faster access to the formal parameters

Disadvantage:

- Recursion is not as useful because values can't be passed

9.4 Suppose you want to write a method that prints a heading on a new output page, along with a page number that is 1 in the first activation and that increases by 1 with each subsequent activation. Can this be done without parameters and without reference to nonlocal variables in Java? Can it be done in C#?

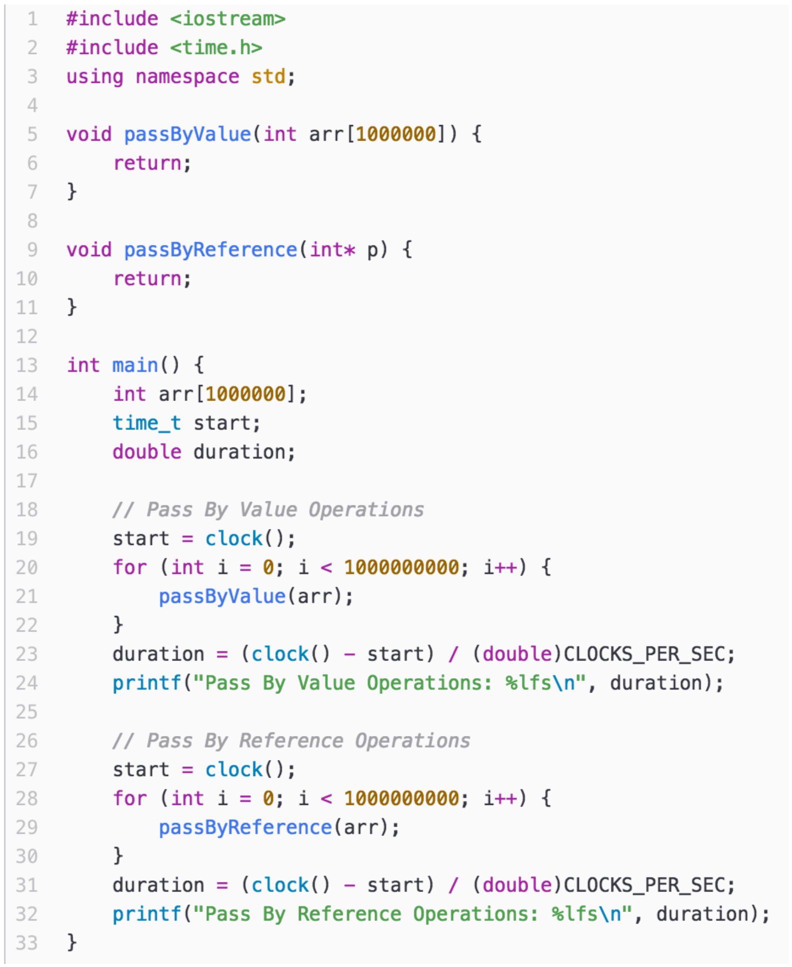
This is achievable in both Java and C# by representing the page number as a static data member.

9.7 Consider the following program written in C syntax. For each of the following parameter-passing methods, what are the values of the *list* array after execution?

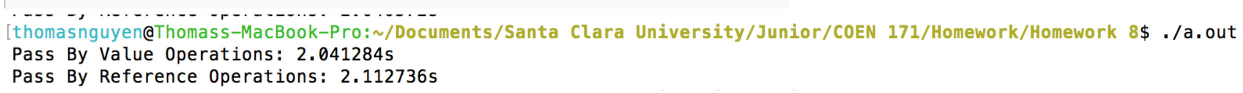
a. **Passed by Value**
   1, 3
b. **Passed by Reference**
   2, 6
c. **Passed by Value-Result**
   2, 6

Programming Exercises

9.1 Write a program in a language that you know to determine the ratio of the time required to pass a large array by reference and the time required to pass the same array by value. Make the array as large as possible on the machine and implementation you use. Pass the array as many times as necessary to get reasonably accurate timings of the passing operations.

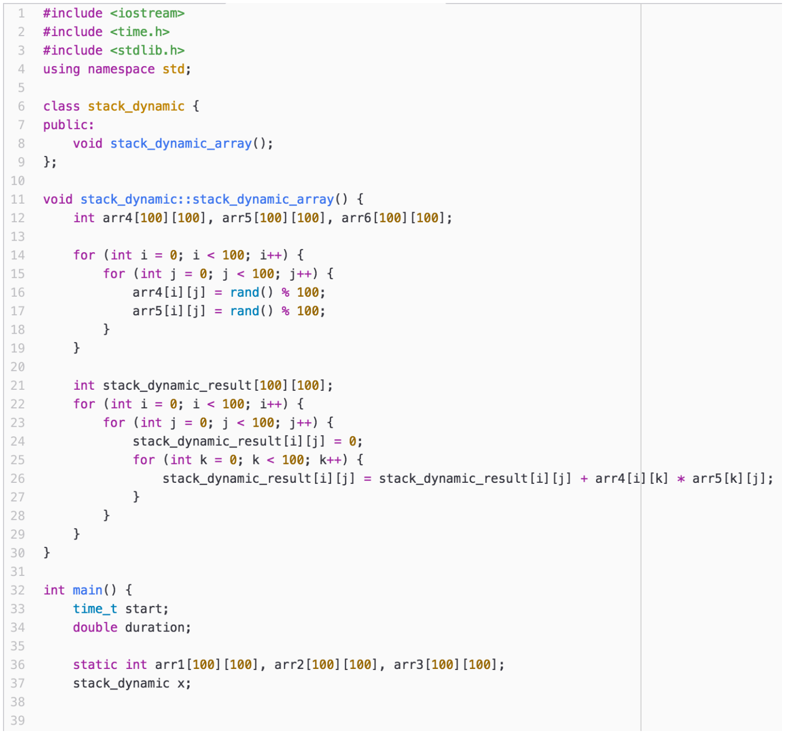


```
#include <iostream>
#include <time.h>
using namespace std;

void passByValue(int arr[1000000]) {
    return;
}

void passByReference(int* p) {
    return;
}

int main() {
    int arr[1000000];
    time_t start;
    double duration;

    // Pass By Value Operations
    start = clock();
    for (int i = 0; i < 1000000000; i++) {
        passByValue(arr);
    }
    duration = (clock() - start) / (double)CLOCKS_PER_SEC;
    printf("Pass By Value Operations: %lfs\n", duration);

    // Pass By Reference Operations
    start = clock();
    for (int i = 0; i < 1000000000; i++) {
        passByReference(arr);
    }
    duration = (clock() - start) / (double)CLOCKS_PER_SEC;
    printf("Pass By Reference Operations: %lfs\n", duration);
}
```

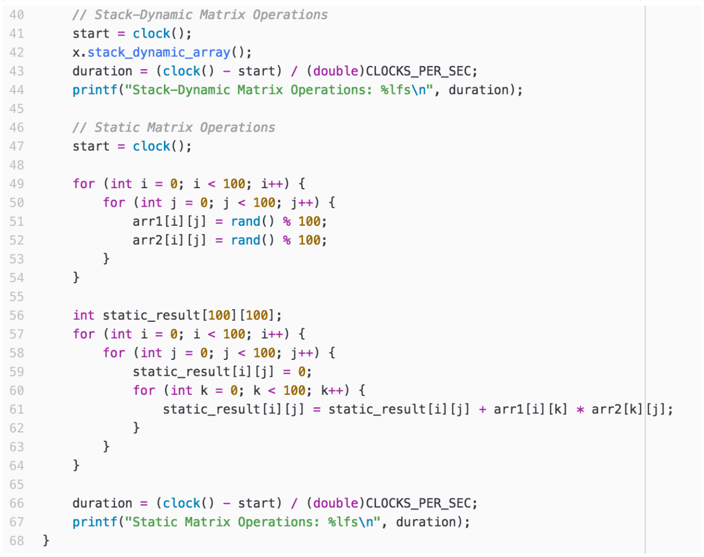


```
[thomasnguyen@Thomass-MacBook-Pro:~/Documents/Santa Clara University/Junior/COEN 171/Homework/Homework 8$ ./a.out
 Pass By Value Operations: 2.041284s
 Pass By Reference Operations: 2.112736s
```

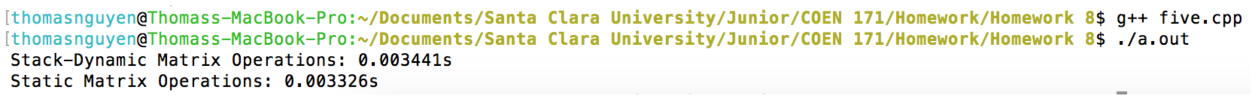
9.5 Write a program in some language that has both static and stack-dynamic local variables in subprograms. Create six large (at least 100 x 100) matrices in the subprogram – three static and three stack dynamic. Fill two of the static matrices and two of the stack-dynamic matrices with random numbers in the range of 1 to 100. The code in the subprogram must perform a large number of matrix multiplication operations on the static matrices and time the process. Then it must repeat this with the stack-dynamic matrices. Compare and explain the results.



```cpp
#include <iostream>
#include <time.h>
#include <stdlib.h>
using namespace std;

class stack_dynamic {
public:
    void stack_dynamic_array();
};

void stack_dynamic::stack_dynamic_array() {
    int arr4[100][100], arr5[100][100], arr6[100][100];

    for (int i = 0; i < 100; i++) {
        for (int j = 0; j < 100; j++) {
            arr4[i][j] = rand() % 100;
            arr5[i][j] = rand() % 100;
        }
    }

    int stack_dynamic_result[100][100];
    for (int i = 0; i < 100; i++) {
        for (int j = 0; j < 100; j++) {
            stack_dynamic_result[i][j] = 0;
            for (int k = 0; k < 100; k++) {
                stack_dynamic_result[i][j] = stack_dynamic_result[i][j] + arr4[i][k] * arr5[k][j];
            }
        }
    }
}

int main() {
    time_t start;
    double duration;

    static int arr1[100][100], arr2[100][100], arr3[100][100];
    stack_dynamic x;

```



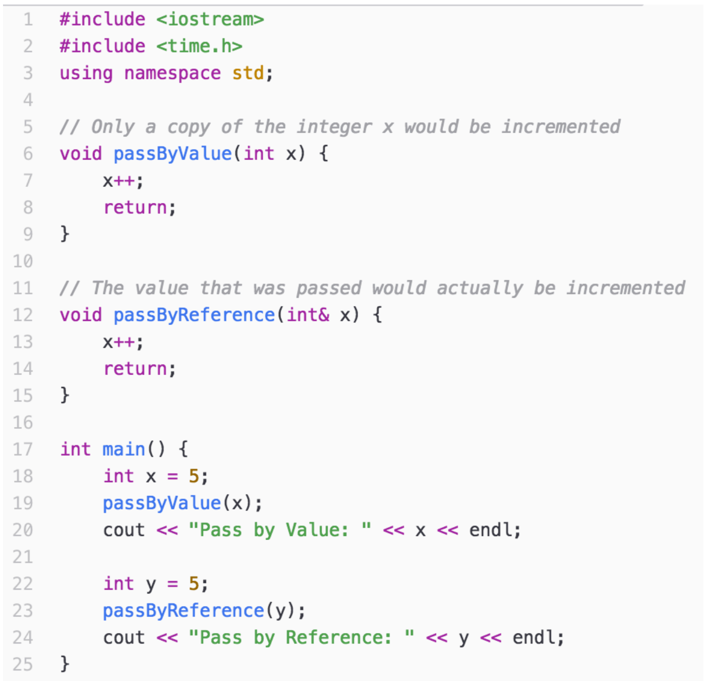
```cpp
    // Stack-Dynamic Matrix Operations
    start = clock();
    x.stack_dynamic_array();
    duration = (clock() - start) / (double)CLOCKS_PER_SEC;
    printf("Stack-Dynamic Matrix Operations: %lfs\n", duration);

    // Static Matrix Operations
    start = clock();

    for (int i = 0; i < 100; i++) {
        for (int j = 0; j < 100; j++) {
            arr1[i][j] = rand() % 100;
            arr2[i][j] = rand() % 100;
        }
    }

    int static_result[100][100];
    for (int i = 0; i < 100; i++) {
        for (int j = 0; j < 100; j++) {
            static_result[i][j] = 0;
            for (int k = 0; k < 100; k++) {
                static_result[i][j] = static_result[i][j] + arr1[i][k] * arr2[k][j];
            }
        }
    }

    duration = (clock() - start) / (double)CLOCKS_PER_SEC;
    printf("Static Matrix Operations: %lfs\n", duration);
}
```



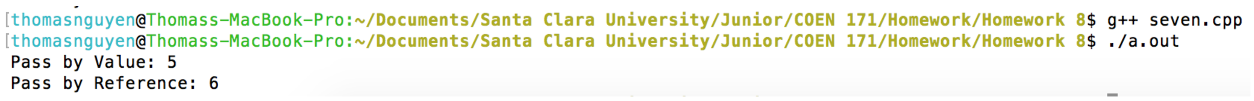
```
[thomasnguyen@Thomass-MacBook-Pro:~/Documents/Santa Clara University/Junior/COEN 171/Homework/Homework 8$ g++ five.cpp
[thomasnguyen@Thomass-MacBook-Pro:~/Documents/Santa Clara University/Junior/COEN 171/Homework/Homework 8$ ./a.out
 Stack-Dynamic Matrix Operations: 0.003441s
 Static Matrix Operations: 0.003326s
```

Stack-dynamic operations take slightly longer than the static matrix operations. This may be because static variables do not need to worry about dynamic allocation and deallocation.

9.7 Write a program, using the syntax of whatever language you like, that produces different behavior depending on whether pass-by-reference or pass-by-value result is used in its parameter passing.



```cpp
#include <iostream>
#include <time.h>
using namespace std;

// Only a copy of the integer x would be incremented
void passByValue(int x) {
    x++;
    return;
}

// The value that was passed would actually be incremented
void passByReference(int& x) {
    x++;
    return;
}

int main() {
    int x = 5;
    passByValue(x);
    cout << "Pass by Value: " << x << endl;

    int y = 5;
    passByReference(y);
    cout << "Pass by Reference: " << y << endl;
}
```



```
[thomasnguyen@Thomass-MacBook-Pro:~/Documents/Santa Clara University/Junior/COEN 171/Homework/Homework 8$ g++ seven.cpp
[thomasnguyen@Thomass-MacBook-Pro:~/Documents/Santa Clara University/Junior/COEN 171/Homework/Homework 8$ ./a.out
 Pass by Value: 5
 Pass by Reference: 6
```