Computer Organization and Architecture

Srivathsa Vamsi Chaturvedula - 22110260

Question -1:

Code:

```cpp
#include <iostream>
#include <ctime>
#include <vector>

using namespace std;

long long fibonacci_recursive(int n) {
    if (n <= 1) return n;
    return fibonacci_recursive(n - 1) + fibonacci_recursive(n - 2);
}

void list_fibonacci_loop(int n) {
    long long fib[50];
    fib[0] = 0;
    fib[1] = 1;
    cout << fib[0] << " " << fib[1] << " ";
    for (int i = 2; i < n; i++) {
        fib[i] = fib[i - 1] + fib[i - 2];
        cout << fib[i] << " ";
    }
    cout << endl;
}

vector<long long> memo(50, -1);
long long fibonacci_memo_recursive(int n) {
    if (memo[n] != -1) return memo[n];
    if (n <= 1) return n;
    memo[n] = fibonacci_memo_recursive(n - 1) + fibonacci_memo_recursive(n - 2);
    return memo[n];
```

```cpp
void list_fibonacci_memo_loop(int n) {
    vector<long long> memo(n, -1);
    memo[0] = 0;
    memo[1] = 1;
    cout << memo[0] << " " << memo[1] << " ";
    for (int i = 2; i < n; i++) {
        memo[i] = memo[i - 1] + memo[i - 2];
        cout << memo[i] << " ";
    }
    cout << endl;
}
```

```cpp
void measure_time_and_print_fibonacci(int n) {
    timespec start, end;
    double time_recursive, time_loop, time_memo_recursive, time_memo_loop;

    clock_gettime(CLOCK_MONOTONIC, &start);
    for (int i = 0; i < n; i++) {
        cout << fibonacci_recursive(i) << " ";
    }
    cout << endl;
    clock_gettime(CLOCK_MONOTONIC, &end);
    time_recursive = (end.tv_sec - start.tv_sec) + (end.tv_nsec - start.tv_nsec) / 1e9;
    cout << "Time taken by recursion: " << time_recursive << " seconds" << endl;

    clock_gettime(CLOCK_MONOTONIC, &start);
    list_fibonacci_loop(n);
    clock_gettime(CLOCK_MONOTONIC, &end);
    time_loop = (end.tv_sec - start.tv_sec) + (end.tv_nsec - start.tv_nsec) / 1e9;
    cout << "Time taken by loop: " << time_loop << " seconds" << endl;

    clock_gettime(CLOCK_MONOTONIC, &start);
    for (int i = 0; i < n; i++) {
        cout << fibonacci_memo_recursive(i) << " ";
    }
    cout << endl;
    clock_gettime(CLOCK_MONOTONIC, &end);
    time_memo_recursive = (end.tv_sec - start.tv_sec) + (end.tv_nsec - start.tv_nsec) / 1e9;
    cout << "Time taken by recursion with memoization: " << time_memo_recursive << " seconds" << endl;

    clock_gettime(CLOCK_MONOTONIC, &start);
    list_fibonacci_memo_loop(n);
    clock_gettime(CLOCK_MONOTONIC, &end);
    time_memo_loop = (end.tv_sec - start.tv_sec) + (end.tv_nsec - start.tv_nsec) / 1e9;
    cout << "Time taken by loop with memoization: " << time_memo_loop << " seconds" << endl;

    cout << "Speedup of loop over recursion: " << time_recursive / time_loop << endl;
    cout << "Speedup of recursion with memoization over recursion: " << time_recursive / time_memo_recursive << endl;
    cout << "Speedup of loop with memoization over recursion: " << time_recursive / time_memo_loop << endl;
}
```

Output:

```
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946 17711 28657 46368 75025 121393 196418 317811 514229 832040 13
46269 2178309 3524578 5702887 9227465 14930352 24157817 39088169 63245986 102334155 165580141 267914296 433494437 701408733 1134903170 1
836311903 2971215073 4807526976 7778742049
Time taken by recursion: 179.005 seconds
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946 17711 28657 46368 75025 121393 196418 317811 514229 832040 13
46269 2178309 3524578 5702887 9227465 14930352 24157817 39088169 63245986 102334155 165580141 267914296 433494437 701408733 1134903170 1
836311903 2971215073 4807526976 7778742049
Time taken by loop: 1.6e-05 seconds
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946 17711 28657 46368 75025 121393 196418 317811 514229 832040 13
46269 2178309 3524578 5702887 9227465 14930352 24157817 39088169 63245986 102334155 165580141 267914296 433494437 701408733 1134903170 1
836311903 2971215073 4807526976 7778742049
Time taken by recursion with memoization: 2.6e-05 seconds
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946 17711 28657 46368 75025 121393 196418 317811 514229 832040 13
46269 2178309 3524578 5702887 9227465 14930352 24157817 39088169 63245986 102334155 165580141 267914296 433494437 701408733 1134903170 1
836311903 2971215073 4807526976 7778742049
Time taken by loop with memoization: 1.7e-05 seconds
Speedup of loop over recursion: 1.11878e+07
Speedup of recursion with memoization over recursion: 6.88481e+06
Speedup of loop with memoization over recursion: 1.05297e+07
```

Question-2:

Code for cpp:

```cpp
#include <iostream>
#include <vector>
#include <ctime>
#include <iomanip>

using namespace std;

// Function to multiply integer matrices
void multiplyIntegerMatrices(const vector<vector<int>>& mat1, const vector<vector<int>>& mat2, vector<vector<int>>& result, int size) {
    for (int i = 0; i < size; ++i) {
        for (int j = 0; j < size; ++j) {
            result[i][j] = 0;
            for (int k = 0; k < size; ++k) {
                result[i][j] += mat1[i][k] * mat2[k][j];
            }
        }
    }
}

// Function to multiply floating-point matrices
void multiplyFloatingPointMatrices(const vector<vector<double>>& mat1, const vector<vector<double>>& mat2, vector<vector<double>>& result, int size) {
    for (int i = 0; i < size; ++i) {
        for (int j = 0; j < size; ++j) {
            result[i][j] = 0.0;
            for (int k = 0; k < size; ++k) {
                result[i][j] += mat1[i][k] * mat2[k][j];
            }
        }
    }
}

// Function to print elapsed time
void printElapsedTime(const struct timespec& start, const struct timespec& end, const string& label) {
    double elapsed = (end.tv_sec - start.tv_sec) * 1e9;
    elapsed = (elapsed + (end.tv_nsec - start.tv_nsec)) * 1e-9;
    cout << label << fixed << elapsed << setprecision(9) << " seconds" << endl;
}
```

```cpp
int main() {
    int sizes[] = {64, 128, 256, 512, 1024};

    for (int size : sizes) {
        struct timespec startSys, endSys, startCpu, endCpu;

        // Record system and CPU start times
        clock_gettime(CLOCK_MONOTONIC, &startSys);
        clock_gettime(CLOCK_PROCESS_CPUTIME_ID, &startCpu);

        vector<vector<int>> matInt1(size, vector<int>(size, 1));
        vector<vector<int>> matInt2(size, vector<int>(size, 1));
        vector<vector<int>> resultInt(size, vector<int>(size, 0));

        struct timespec startMult, endMult;

        clock_gettime(CLOCK_MONOTONIC, &startMult);

        // Perform integer matrix multiplication
        multiplyIntegerMatrices(matInt1, matInt2, resultInt, size);

        clock_gettime(CLOCK_MONOTONIC, &endMult);

        clock_gettime(CLOCK_PROCESS_CPUTIME_ID, &endCpu);
        clock_gettime(CLOCK_MONOTONIC, &endSys);

        cout << "Size=" << size << ", Integer Matrix Multiplication: " << endl;
        printElapsedTime(startSys, endSys, "System Time: ");
        printElapsedTime(startCpu, endCpu, "CPU Time: ");
        printElapsedTime(startMult, endMult, "Multiplication Time: ");

        double propMultSys = (endMult.tv_sec - startMult.tv_sec) * 1e9;
        propMultSys = (propMultSys + (endMult.tv_nsec - startMult.tv_nsec)) * 1e-9;
        double systemTime = (endSys.tv_sec - startSys.tv_sec) * 1e9;
        systemTime = (systemTime + (endSys.tv_nsec - startSys.tv_nsec)) * 1e-9;
        cout << "Proportion (Multiplication/System): " << fixed << setprecision(9) << (propMultSys / systemTime) << endl;
```

```
        clock_gettime(CLOCK_MONOTONIC, &startSys);
        clock_gettime(CLOCK_PROCESS_CPUTIME_ID, &startCpu);

        vector<vector<double>> matFloat1(size, vector<double>(size, 1.0));
        vector<vector<double>> matFloat2(size, vector<double>(size, 1.0));
        vector<vector<double>> resultFloat(size, vector<double>(size, 0.0));

        clock_gettime(CLOCK_MONOTONIC, &startMult);

        // Perform floating-point matrix multiplication
        multiplyFloatingPointMatrices(matFloat1, matFloat2, resultFloat, size);

        clock_gettime(CLOCK_MONOTONIC, &endMult);

        clock_gettime(CLOCK_PROCESS_CPUTIME_ID, &endCpu);
        clock_gettime(CLOCK_MONOTONIC, &endSys);

        cout << "Size=" << size << ", Floating-Point Matrix Multiplication: " << endl;
        printElapsedTime(startSys, endSys, "System Time: ");
        printElapsedTime(startCpu, endCpu, "CPU Time: ");
        printElapsedTime(startMult, endMult, "Multiplication Time: ");

        propMultSys = (endMult.tv_sec - startMult.tv_sec) * 1e9;
        propMultSys = (propMultSys + (endMult.tv_nsec - startMult.tv_nsec)) * 1e-9;
        systemTime = (endSys.tv_sec - startSys.tv_sec) * 1e9;
        systemTime = (systemTime + (endSys.tv_nsec - startSys.tv_nsec)) * 1e-9;
        cout << "Proportion (Multiplication/System): " << fixed << setprecision(9) << (propMultSys / systemTime) << endl;
    }

    return 0;
}
```

Time Calculations for CPP:

Dimension	Matrix Type	System Time (s)	CPU Time (s)	Multiplication Time (s)	Proportion (Multiplication/System)
64	Integer	0.004612	0.015625	0.0045216	0.980483997
64	Floating-Point	0.0028134	0.000000	0.0026935	0.957382526
128	Integer	0.0315005	0.031250	0.0313255	0.994444533
128	Floating-Point	0.024052	0.000000	0.0237494	0.987418926
256	Integer	0.1914277	0.125000	0.1910465	0.998008648
256	Floating-Point	0.2060596	0.062500	0.2053793	0.996698528
512	Integer	1.6230697	1.015625	1.6219184	0.999290665
512	Floating-Point	1.8347737	1.437500	1.8324665	0.998742515
1024	Integer	14.8837908	10.921875	14.8782566	0.999628173
1024	Floating-Point	20.0175612	13.093750	20.0057027	0.999407595

Code in python:

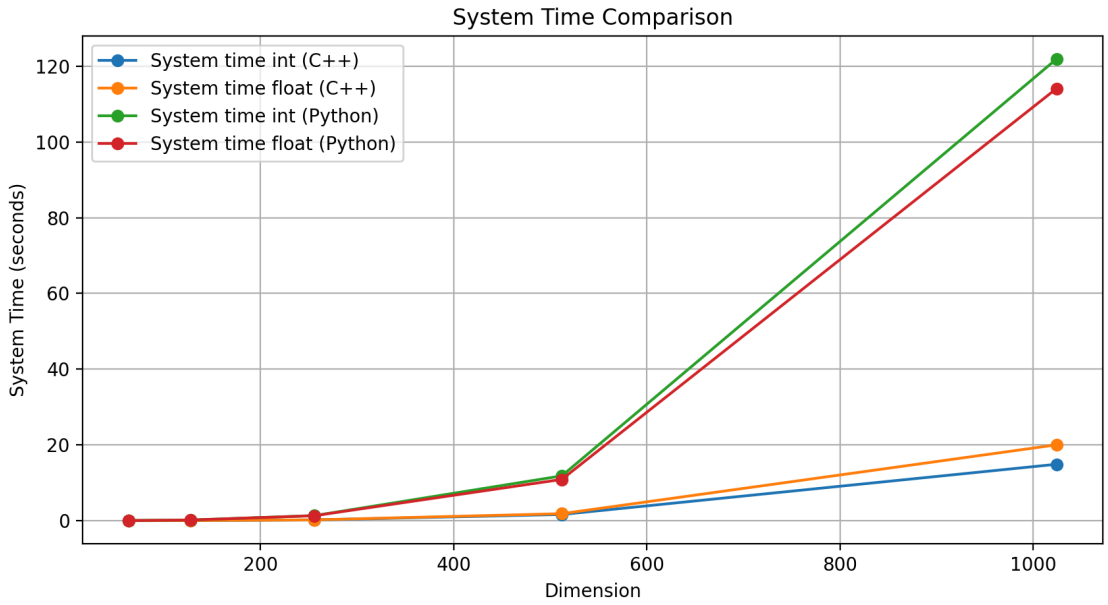
```python
import time

def multiply_matrices_int(matrix_a, matrix_b, n):
    result = [[0 for _ in range(n)] for _ in range(n)]
    for i in range(n):
        for j in range(n):
            for k in range(n):
                result[i][j] += matrix_a[i][k] * matrix_b[k][j]
    return result

def multiply_matrices_double(matrix_a, matrix_b, n):
    result = [[0.0 for _ in range(n)] for _ in range(n)]
    for i in range(n):
        for j in range(n):
            for k in range(n):
                result[i][j] += matrix_a[i][k] * matrix_b[k][j]
    return result

sizes = [64, 128, 256, 512, 1024]

for n in sizes:
    # Integer matrix multiplication timing
    start_sys_time = time.time()
    start_cpu_time = time.process_time()
    int_matrix_a = [[1 for _ in range(n)] for _ in range(n)]
    int_matrix_b = [[1 for _ in range(n)] for _ in range(n)]
    mult_start_time = time.time()
    int_result_matrix = multiply_matrices_int(int_matrix_a, int_matrix_b, n)
    mult_end_time = time.time()
    end_cpu_time = time.process_time()
    end_sys_time = time.time()

    print(f"Size={n}, Integer Matrix Multiplication:")
    print(f"System Time: {end_sys_time - start_sys_time:.9f} seconds")
    print(f"CPU Time: {end_cpu_time - start_cpu_time:.9f} seconds")
    print(f"Multiplication Time: {mult_end_time - mult_start_time:.9f} seconds")
```

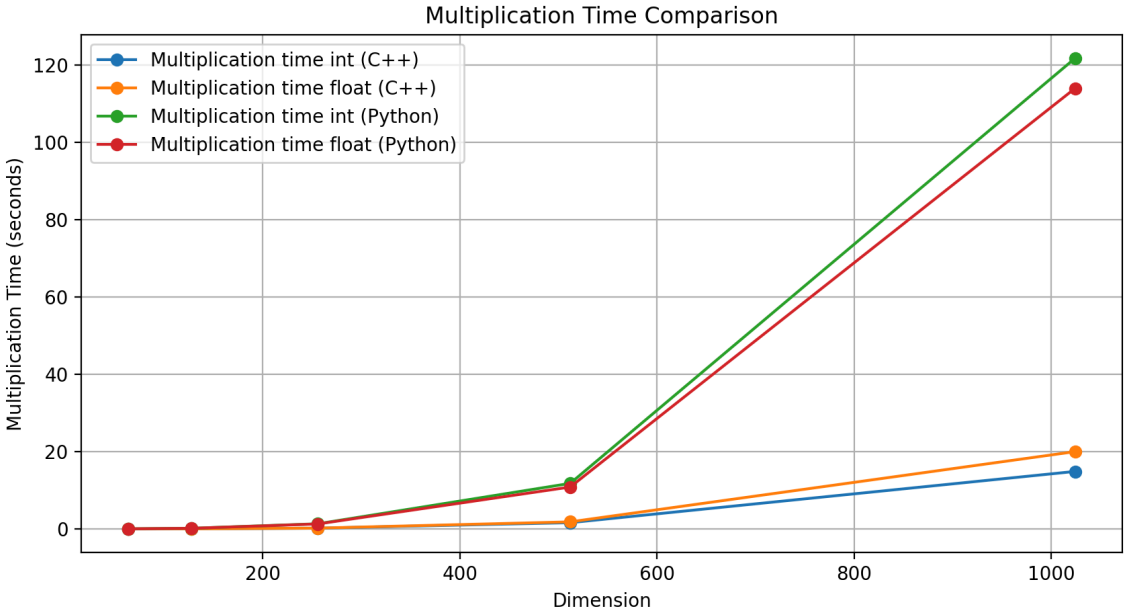
```python
# Double matrix multiplication timing
start_sys_time = time.time()
start_cpu_time = time.process_time()
double_matrix_a = [[1.0 for _ in range(n)] for _ in range(n)]
double_matrix_b = [[1.0 for _ in range(n)] for _ in range(n)]
mult_start_time = time.time()
double_result_matrix = multiply_matrices_double(double_matrix_a, double_matrix_b, n)
mult_end_time = time.time()
end_cpu_time = time.process_time()
end_sys_time = time.time()

print(f"Size={n}, Double Matrix Multiplication:")
print(f"System Time: {end_sys_time - start_sys_time:.9f} seconds")
print(f"CPU Time: {end_cpu_time - start_cpu_time:.9f} seconds")
print(f"Multiplication Time: {mult_end_time - mult_start_time:.9f} seconds")
```

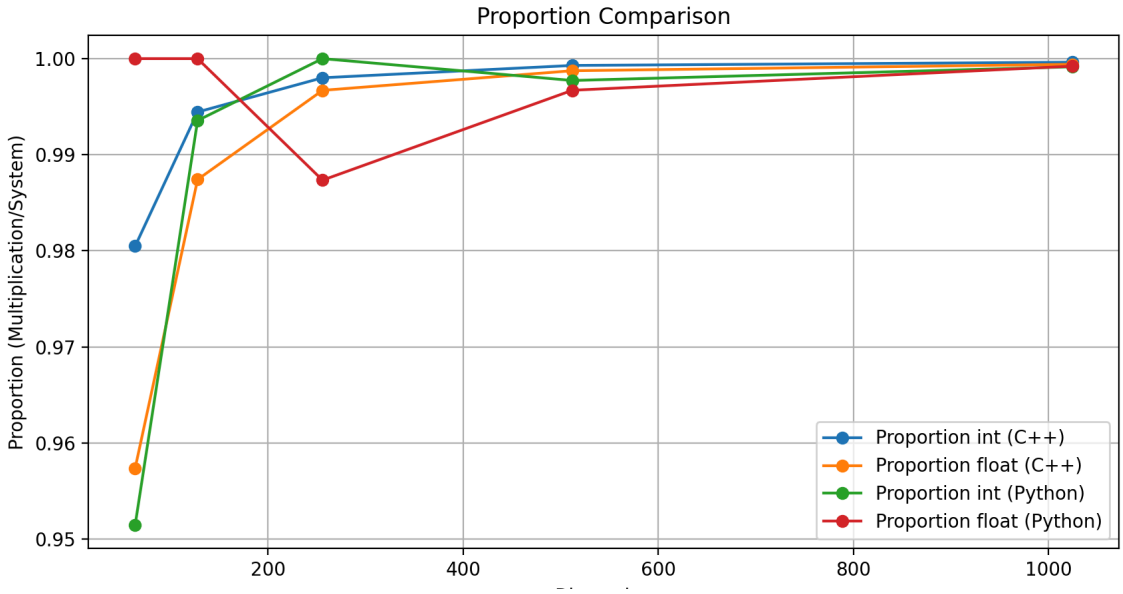
Time Calculations for Python:

Size	Type	System Time (seconds)	CPU Time (seconds)	Multiplication Time (seconds)	Proportion (Multiplication/System)
64	Integer	0.022954	0.015625	0.02184	0.951468154
64	Double	0.020769	0	0.020769	1
128	Integer	0.17063	0.125	0.169531	0.993559163
128	Double	0.163134	0.140625	0.163134	1
256	Integer	1.32919	1.09375	1.32919	1
256	Double	1.302475	1.140625	1.286003	0.987353308
512	Integer	11.820492	11.75	11.793606	0.997725475
512	Double	10.883227	10.828125	10.84738	0.996706216
1024	Integer	121.910728	115.1875	121.807019	0.999149304
1024	Double	114.112407	104.875	114.023086	0.999217254

Plot representations







(All Values are almost equal to 1 in Proportion)