|  |
| --- |
| University School of Information, Communication and Technology |
| Laboratory Manual |
| IT206: Design and Analysis of Algorithms |

|  |
| --- |
| Prof. Pravin Chandra  March, 2023 |

Table of Contents

[Introduction 1](#_Toc137036387)

[Lab001: Implement a timer and a random number generator class. 2](#_Toc137036388)

[LAB002: Implementation of Insertion sort 5](#_Toc137036389)

[LAB003: Implementation of Bubble sort 9](#_Toc137036390)

[LAB004: Implementation of Selection sort 10](#_Toc137036391)

[LAB005: Implementation of Shell sort 11](#_Toc137036392)

[LAB006: Implementation of Quicksort 12](#_Toc137036393)

[LAB007: Implementation of Heapsort 13](#_Toc137036394)

[LAB008: Implementation of Mergesort 14](#_Toc137036395)

[LAB009: Implementation of Randomized Quicksort 15](#_Toc137036396)

[Declaration of Comparison based Sorting Techniques 16](#_Toc137036397)

[EXTRA001: Empirical Comparison of Sorting Algorithms (Extra Assignment) 17](#_Toc137036398)

[LAB010: Implementation of Counting sort 18](#_Toc137036399)

[LAB011: Implementation of Radix sort 20](#_Toc137036400)

[LAB012: Implementation of Bucket sort 21](#_Toc137036401)

[Definitions of the non-comparison based sorting techniques 22](#_Toc137036402)

[EXTRA002: Implementation of Bucket sort for Integers (Extra Assignment) 23](#_Toc137036403)

[EXTRA003: Implementation of Strassen’s Algorithm for Multiplication of Compatible Matrices (Extra Assignment) 24](#_Toc137036404)

[LAB013: Implement the Min, Max, MinMax and the Select ith smallest element algorithm 25](#_Toc137036405)

[Lab014: Implementation of an unordered list. 34](#_Toc137036406)

[Lab015: Implementation of a data structure and algorithms for Sets. 40](#_Toc137036407)

[Lab016: Implementation of a data structure and algorithms for Disjoint Sets. 44](#_Toc137036408)

[Lab017: Implementation of Disjoint Set Forest (data structure and algorithms). 49](#_Toc137036409)

[Lab018: Implementation of a Matrix class. 52](#_Toc137036410)

[Lab019: Implementation of a Matrix Chain Multiplication Algorithm(s). 60](#_Toc137036411)

[Lab020: Longest Common Subsequence 65](#_Toc137036412)

[Lab021: Generation of all possible BST if keys are 1 to N 71](#_Toc137036413)

[Lab022: Generation of all possible BST if Keys and associate probabilities are given. 75](#_Toc137036414)

[Lab023: 0/1 Knap sack problem with exhaustive enumeration of feasible subsets of items 82](#_Toc137036415)

[Lab024: Solve the 0/1 Knap sack problem using dynamic programming 94](#_Toc137036416)

[LAB025: Write a program to find the solution of the fractional Knapsack problem. 99](#_Toc137036417)

[LAB026: Implement Huffman’s Encoding Algorithm using a tree 100](#_Toc137036418)

[LAB027: Implement Huffman’s Encoding Algorithm using a Min-Heap 102](#_Toc137036419)

[LAB028: Implement the graph Abstract Data Type using an adjacency matrix (undirected graphs). 103](#_Toc137036420)

[LAB029: Implement the graph Abstract Data Type using an adjacency matrix (directed graphs). 105](#_Toc137036421)

[LAB030: Implement the graph Abstract Data Type using an adjacency list (undirected graphs). 106](#_Toc137036422)

[LAB031: Implement the graph Abstract Data Type using an adjacency list (directed graphs). 107](#_Toc137036423)

[LAB032: Implement the Breadth First Search algorithm on a graph (adjacency matrix representation). 108](#_Toc137036424)

[LAB033: Implement the Breadth First Search algorithm on a graph (adjacency list representation). 109](#_Toc137036425)

[LAB034: Implement the Depth First Search algorithm on a graph (adjacency matrix representation). 110](#_Toc137036426)

[LAB035: Implement the Depth First Search algorithm on a graph (adjacency list representation). 111](#_Toc137036427)

[LAB036: Implement the Topological Sort algorithm on a graph (adjacency matrix representation). 112](#_Toc137036428)

[LAB037: Implement the Topological Sort algorithm on a graph (adjacency list representation). 113](#_Toc137036429)

[LAB038: Implement the Strongly Connected Component algorithm on a graph (adjacency matrix representation). 114](#_Toc137036430)

[LAB039: Implement the Strongly Connected Component algorithm on a graph (adjacency list representation). 115](#_Toc137036431)

[LAB040: Implement the Kruskal’s and Prim’s minimum spanning tree algorithm on a graph (adjacency matrix representation). 116](#_Toc137036432)

[LAB041: Implement the Kruskal’s and Prim’s minimum spanning tree algorithm on a graph (adjacency list representation). 117](#_Toc137036433)

[LAB042: Implement the Bellman-Ford’s and Dijikstra’s single source shortest path algorithm on a graph (adjacency matrix representation). 118](#_Toc137036434)

[LAB043: Implement the Bellman-Ford’s and Dijikstra’s single source shortest path algorithm on a graph (adjacency list representation). 119](#_Toc137036435)

[LAB044: Implement the Flloyd-Warshall’s all pair shortest path algorithm on a graph (adjacency matrix representation). 120](#_Toc137036436)

[LAB045: Implement the Flloyd-Warshall’s all pair shortest path algorithm on a graph (adjacency list representation). 121](#_Toc137036437)

[LAB046: Implement the naïve (sub) string matching algorithm. 122](#_Toc137036438)

[LAB047: Implement the Rabin Karp (sub)string matching algorithm. 123](#_Toc137036439)

**Index: UNIT I**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Lab.** | **Nature of assignment** | **Due Date\*** | **Submitted on** | **Signature** |
| LAB001 | Compulsory | 20 April |  |  |
| LAB002 | Optional | 20 April |  |  |
| LAB003 | Optional | 20 April |  |  |
| LAB004 | Optional | 20 April |  |  |
| LAB005 | Optional | 20 April |  |  |
| LAB006 | Compulsory | 20 April |  |  |
| LAB007 | Compulsory | 20 April |  |  |
| LAB008 | Compulsory | 20 April |  |  |
| LAB009 | Compulsory | 20 April |  |  |
| EXTRA001 | Extra Assignment | No Date |  |  |
| LAB010 | Compulsory | 27 April |  |  |
| LAB011 | Compulsory | 27 April |  |  |
| LAB012 | Compulsory | 27 April |  |  |
| EXTRA002 | Extra Assignment | No Date |  |  |
| EXTRA003 | Extra Assignment | No Date |  |  |
| LAB013 | Compulsory | 27 April |  |  |
| LAB014 | Compulsory | 27 April |  |  |
| LAB015 | Compulsory | 27 April |  |  |
| LAB016 | Compulsory | 27 April |  |  |
| LAB017 | Compulsory | 27 April |  |  |

\* On or before

Note: 13 Compulsory laboratory assignments while the rest are optional (in all 20).

**Index: UNIT II**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Lab.** | **Nature of assignment** | **Due Date\*** | **Submitted on** | **Signature** |
| LAB018 | Compulsory | 4th May |  |  |
| LAB019 | Compulsory | 4th May |  |  |
| LAB020 | Compulsory | 4th May |  |  |
| LAB021 | Optional | 4th May |  |  |
| LAB022 | Compulsory | 4th May |  |  |
| LAB023 | Compulsory | 18th May |  |  |
| LAB024 | Compulsory | 18th May |  |  |
| LAB025 | Compulsory | 18th May |  |  |
| LAB026 | Optional | 18th May |  |  |
| LAB027 | Compulsory | 18th May |  |  |

\* On or before

Note: 08 Compulsory laboratory assignments while the rest are optional (in all 10).

**Index: UNIT III**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Lab.** | **Nature of assignment** | **Due Date\*** | **Submitted on** | **Signature** |
| LAB028 | Compulsory | 15th June |  |  |
| LAB029 | Optional | 15th June |  |  |
| LAB030 | Compulsory | 15th June |  |  |
| LAB031 | Optional | 15th June |  |  |
| LAB032 | Compulsory | 15th June |  |  |
| LAB033 | Optional | 15th June |  |  |
| LAB034 | Compulsory | 15th June |  |  |
| LAB035 | Optional | 15th June |  |  |
| LAB036 | Compulsory | 15th June |  |  |
| LAB037 | Optional | 15th June |  |  |
| LAB038 | Compulsory | 15th June |  |  |
| LAB039 | Optional | 15th June |  |  |
| LAB040 | Compulsory | 15th June |  |  |
| LAB041 | Optional | 15th June |  |  |
| LAB042 | Compulsory | 15th June |  |  |
| LAB043 | Optional | 15th June |  |  |
| LAB044 | Compulsory | 15th June |  |  |
| LAB045 | Optional | 15th June |  |  |

\* On or before

Note: 09 Compulsory laboratory assignments while the rest are optional (in all 18).

**Index: UNIT IIa**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Lab.** | **Nature of assignment** | **Due Date\*** | **Submitted on** | **Signature** |
| LAB046 | Compulsory | 15th June |  |  |
| LAB047 | Compulsory | 15th June |  |  |

# Introduction

The laboratory work is divided into experiments. All experiments are to be performed together with the deadline specified. The instructions for each laboratory experiment is defined. Late submissions shall be penalized and marked as zero. Assignments due on the same date shall be evaluated as a group. An experiment may be marked as EXTRA to represent that they are optional and shall be considered for extra credits / marks in the course. Copying of code is not allowed. All submissions should be in a MS Word .docx file as well as printouts. The index of files must be submitted for signature and evaluation in printed form.

# Lab001: Implement a timer and a random number generator class.

The timer class is implemented as:

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//FILE: myTimer.h

#ifndef MYTIMER\_H\_INCLUDED

#define MYTIMER\_H\_INCLUDED

#include <windows.h>

class myTimer{

LARGE\_INTEGER Frequency;

LARGE\_INTEGER startTime;

LARGE\_INTEGER endTime;

double interval;

public:

myTimer() { QueryPerformanceFrequency(&Frequency); }

void StartTimer(){ QueryPerformanceCounter(&startTime); }

void EndTimer(){ QueryPerformanceCounter(&endTime); }

double GetInterval() {

return (double) (endTime.QuadPart - startTime.QuadPart) / Frequency.QuadPart;

}

};

#endif // MYTIMER\_H\_INCLUDED

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

The random number generator class is :

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//FILE: myRandom.h

#ifndef MYRANDOM\_H\_INCLUDED

#define MYRANDOM\_H\_INCLUDED

class myRandom{

static const unsigned long a = 1664525L, c = 1664525L;

static unsigned long x;

public:

myRandom(unsigned long s=8462817L) { x = s;}

static void setSeed(unsigned long s){ x = s;}

static unsigned long getVal() { return x = a\*x + c;}

static unsigned long getCurVal(){ return x;}

};

unsigned long myRandom::x = 8462817L;

#endif // MYRANDOM\_H\_INCLUDED

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Write a main function to generate files containing N number of random long integers as:

|  |  |
| --- | --- |
| **File Name** | **N** |
| NUM-01-000005.txt, NUM-02-000005.txt, … , NUM-30-000005.txt | 5 |
| NUM-01-000010.txt, NUM-02-000010.txt, … , NUM-30-000010.txt | 10 |
| NUM-01-000025.txt, NUM-02-000025.txt, … , NUM-30-000025.txt | 25 |
| NUM-01-000050.txt, NUM-02-000050.txt, … , NUM-30-000050.txt | 50 |
| NUM-01-000100.txt, NUM-02-000100.txt, … , NUM-30-000100.txt | 100 |
| NUM-01-000250.txt, NUM-02-000250.txt, … , NUM-30-000250.txt | 250 |
| NUM-01-000500.txt, NUM-02-000500.txt, … , NUM-30-000500.txt | 500 |
| NUM-01-001000.txt, NUM-02-001000.txt, … , NUM-30-001000.txt | 1000 |
| NUM-01-002500.txt, NUM-02-002500.txt, … , NUM-30-002500.txt | 2500 |
| NUM-01-005000.txt, NUM-02-005000.txt, … , NUM-30-005000.txt | 5000 |
| NUM-01-010000.txt, NUM-02-010000.txt, … , NUM-30-010000.txt | 10000 |
| NUM-01-025000.txt, NUM-02-025000.txt, … , NUM-30-025000.txt | 25000 |
| NUM-01-050000.txt, NUM-02-050000.txt, … , NUM-30-050000.txt | 50000 |
| NUM-01-100000.txt, NUM-02-100000.txt, … , NUM-30-100000.txt | 100000 |

For these files also find the average time taken for generation. The syntax for usage should be as:

LAB001 <N> <M> <S>

The command line should minimally specify the ‘N’ value, which the value of number of random values per file and ‘M’ value which is the number of files to be generated containing ‘N’ random integers. ‘S’ is an optional parameter specifying the initial random seed value. N, M and S (if present) are assumed to be unsigned long integers

The output should be something like this:

LAB001 5 30

Average Time Taken: 0.00072 seconds

LAB001 10 30

Average Time Taken: 0.00081 seconds

LAB001 25 30

Average Time Taken: 0.00114 seconds

LAB001 50 30

Average Time Taken: 0.00172 seconds

LAB001 100 30

Average Time Taken: 0.00661 seconds

LAB001 250 30

Average Time Taken: 0.00363 seconds

LAB001 500 30

Average Time Taken: 0.00705 seconds

LAB001 1000 30

Average Time Taken: 0.01475 seconds

LAB001 2500 30

Average Time Taken: 0.02093 seconds

LAB001 5000 30

Average Time Taken: 0.04223 seconds

LAB001 10000 30

Average Time Taken: 0.06743 seconds

LAB001 25000 30

Average Time Taken: 0.18210 seconds

LAB001 50000 30

Average Time Taken: 0.35463 seconds

LAB001 100000 30

Average Time Taken: 0.68953 seconds

Note: Write your own function to convert a string into an unsigned long integer. The declaration for this function should be in a header file as:

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//FILE: myAuxFunctions.h

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#ifndef MYAUXFUNCTIONS\_H\_INCLUDED

#define MYAUXFUNCTIONS\_H\_INCLUDED

unsigned long StrToULong(char\* x);

#endif // MYAUXFUNCTIONS\_H\_INCLUDED

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

The driver code is:

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//FILE: main.cpp

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <iostream>

#include <fstream>

#include <sstream>

#include <string>

#include <iomanip>

#include "myTimer.h"

#include "myAuxFunctions.h"

#include "myRandom.h"

using namespace std;

int main(int argc, char \*argv[]){

if (argc < 3) {

cout << "Syntax:\n\tLAB001 <N> <M> <S>"

<< "\n\twhere N is an integer representing the number "

<< "\n\tof random numbers to be generated per file while"

<< "\n\tM is the number of files to be generated. "

<< "\n\n\tWhereas S is an optional parameter specifing "

<< "\n\tthe initial seed of the random number generator.\n\n";

return 0;

}

//START MEASURING TIME FROM HERE

myTimer T;

T.StartTimer();

//////////////////////////////////////////////////////////

myRandom R;

unsigned long N = StrToULong(argv[1]), M = StrToULong(argv[2]);

if (argc == 4) R.setSeed(StrToULong(argv[3]));

//print the commandline

for (int i = 0; i < argc; i++) cout << argv[i] << " ";

cout << endl;

string fileName;

for (int i = 0; i < int(M); i++){

ostringstream strStream;

strStream << "NUM-" << setw(2) << setfill('0') << to\_string(i+1)

<< "-" << setw(6) << setfill('0') << to\_string(N) << ".txt";

fileName = strStream.str();

//cout << fileName << endl;

ofstream OUTFILE(fileName,ios::out | ios::trunc);

if (!OUTFILE) {

cout << "Could not open file: " << fileName << endl;

return 0;

}

for (int j = 0; j < N; j++) {

OUTFILE << R.getVal() << endl;

}

OUTFILE.close();

}

///////////////////////END THE TIME

T.EndTimer();

cout << "Average Time Taken: "

<< setw(20) << setprecision(10) << T.GetInterval() / M << " seconds" << endl;

return 0;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**Attach your code here**

# LAB002: Implementation of Insertion sort

The aim is to implement Insertion sort in its standard implementation. The data to be used is the same as generated in the LAB001. Find the actual process time taken for each sized data (on an average and find the best fit regression equation for the same). Also find the number of comparisons for each sized data (on an average and find the best fit regression equation for the same).

The syntax of all the function declarations for comparison based sorting is after LAB009 in the file **mySort.h**. The driver code is:

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// FILE: main.cpp

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <iostream>

#include <fstream>

#include <sstream>

#include <string>

#include <iomanip>

#include "myTimer.h"

#include "mySort.h"

#define NUMBER\_OF\_SIZES 14

#define NUMBER\_OF\_SAMPLES 30

using namespace std;

void Analyse(double timeTaken[NUMBER\_OF\_SIZES][NUMBER\_OF\_SAMPLES],

unsigned long COUNT[NUMBER\_OF\_SIZES][NUMBER\_OF\_SAMPLES],

double AvgTime[NUMBER\_OF\_SIZES], double AvgCount[NUMBER\_OF\_SIZES]);

int main() {

unsigned long N[] = {5,10,25, 50,100,250, 500,1000,2500, 5000,10000,25000, 50000,100000};

double timeTaken[NUMBER\_OF\_SIZES][NUMBER\_OF\_SAMPLES];

unsigned long COUNT[NUMBER\_OF\_SIZES][NUMBER\_OF\_SAMPLES];

string fileName;

myTimer T;

for (int j = 0; j < NUMBER\_OF\_SIZES; j++) {

unsigned long Data[N[j]];

//cout << "N = " << setw(10)<< N[j] << " ";

for (int i = 0; i < NUMBER\_OF\_SAMPLES; i++){

ostringstream strStream;

strStream << "NUM-" << setw(2) << setfill('0') << to\_string(i+1)

<< "-" << setw(6) << setfill('0') << to\_string(N[j]) << ".txt";

fileName = strStream.str();

cout << fileName << endl;

ifstream INFILE(fileName);

if (!INFILE) {

cout << "Could not open file: " << fileName << endl;

return 0;

}

for (unsigned long k = 0; k < N[j]; k++) INFILE >> Data[k];

INFILE.close();

T.StartTimer();

COUNT[j][i]=InsertionSort(Data,N[j]);

T.EndTimer();

timeTaken[j][i] = T.GetInterval();

//cout << setw(20) << setprecision(10) << timeTaken[j][i] << " ";

}

//cout << endl;

}

double AvgTime[NUMBER\_OF\_SIZES],AvgCount[NUMBER\_OF\_SIZES];

Analyse(timeTaken,COUNT,AvgTime, AvgCount);

cout << setw(6) << "N" << " "

<< setw(15) << "Average Time" << " "

<< setw(15) << "Average Count" << endl;

for (int i = 0; i < NUMBER\_OF\_SIZES; i++ ){

cout << setw(6) << N[i] << " "

<< setw(15) << AvgTime[i] << " "

<< setw(15) << AvgCount[i] << endl;

}

ofstream OUTFILE("OUTPUT.txt",ios::out | ios::trunc);

if (!OUTFILE) {

cout << "Could not open file: OUTPUT.txt" << endl;

return 0;

}

OUTFILE << setw(6) << "N" << " "

<< setw(15) << "Average Time" << " "

<< setw(15) << "Average Count" << endl;

for (int i = 0; i < NUMBER\_OF\_SIZES; i++ ){

OUTFILE << setw(6) << N[i] << " "

<< setw(15) << AvgTime[i] << " " << setw(15) << AvgCount[i] << endl;

}

OUTFILE.close();

return 0;

}

void Analyse(double timeTaken[NUMBER\_OF\_SIZES][NUMBER\_OF\_SAMPLES],

unsigned long COUNT[NUMBER\_OF\_SIZES][NUMBER\_OF\_SAMPLES],

double AvgTime[NUMBER\_OF\_SIZES], double AvgCount[NUMBER\_OF\_SIZES])

{

for (int i = 0; i < NUMBER\_OF\_SIZES; i++){

AvgTime[i] = 0.0;

AvgCount[i] = 0;

for (int j = 0; j < NUMBER\_OF\_SAMPLES; j++) {

AvgTime[i] += timeTaken[i][j];

AvgCount[i] += COUNT[i][j];

}

AvgTime[i] /= double(NUMBER\_OF\_SAMPLES);

AvgCount[i] /= double(NUMBER\_OF\_SAMPLES);

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**Describe the algorithm, establish its correctness and analyse its timing (best case, worst case and average case)**

The output should be something like:

N Average Time Average Count

5 2.53333e-07 4.96667

10 4.56667e-07 24.2333

25 1.53e-06 148.5

50 4.10667e-06 609.567

100 1.32967e-05 2472.87

250 5.428e-05 15435.6

500 0.000187293 62426.2

1000 0.000701753 249138

2500 0.00471364 1.56359e+06

5000 0.0177953 6.233e+06

10000 0.0805101 2.49947e+07

25000 0.458194 1.56368e+08

50000 1.89531 6.24792e+08

100000 6.98623 2.50021e+09

The curves to be plotted are something like:

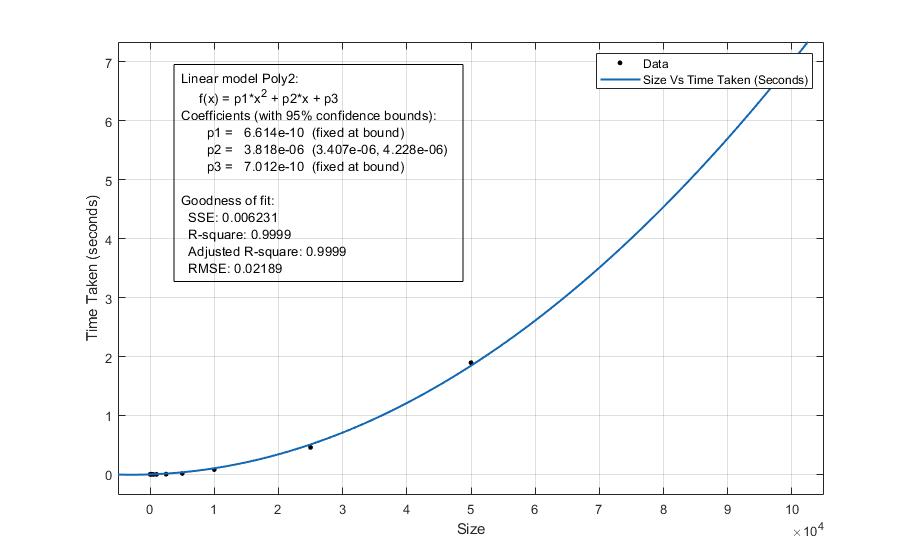


Fig 1: Plot of size of samples vs time taken.

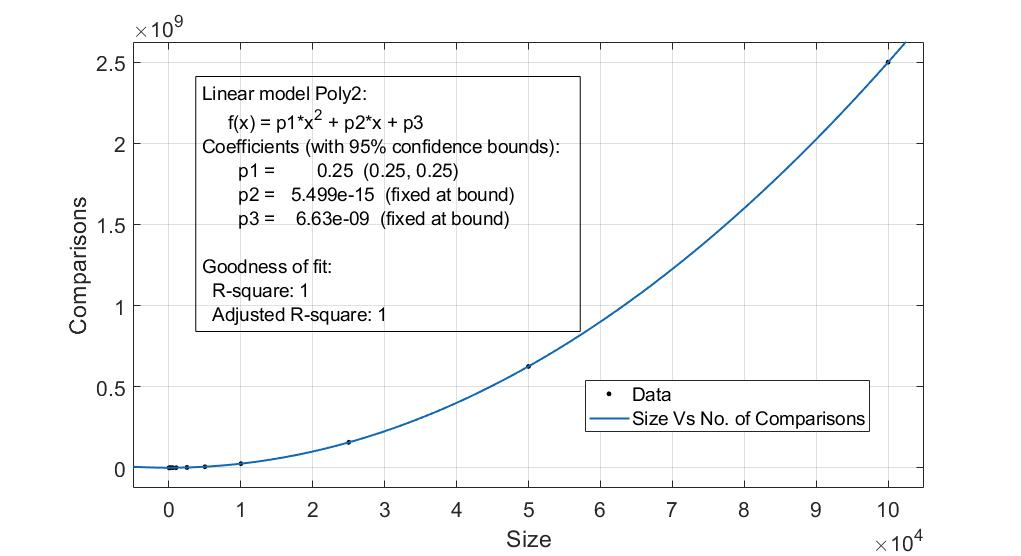


Fig 2: Plot of Size vs the number of comparisons

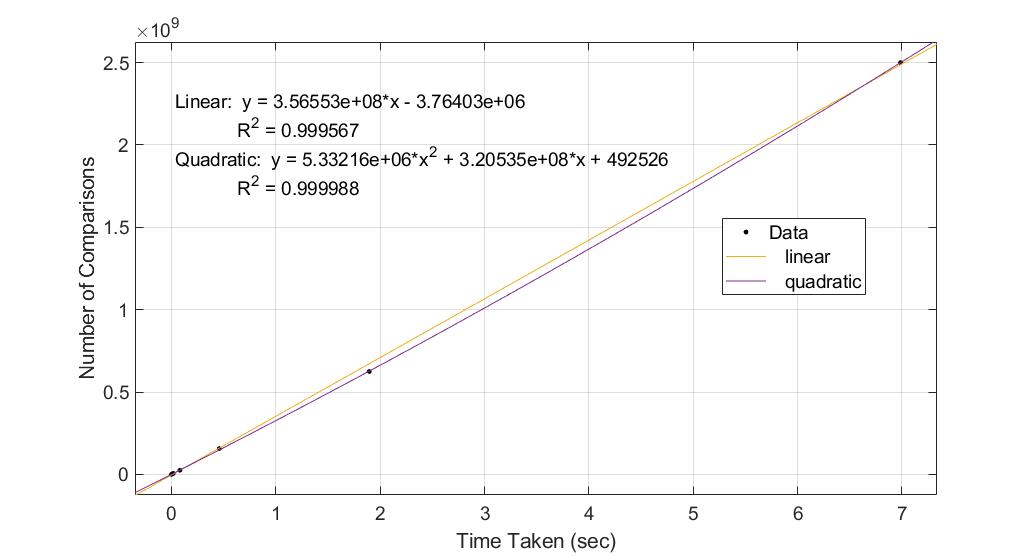


Fig 3: Plot of Time Taken (Sec) vs Number of Comparisons.

**Summarize your findings**

**Attach your code here.**

**Note: the syntax for the implementation is after LAB009 description for all comparison based sorting experiments**

# LAB003: Implementation of Bubble sort

The aim is to implement bubble sort in its standard implementation. The data to be used is the same as generated in the LAB001. Find the actual process time taken for each sized data (on an average and find the best fit regression equation for the same). Also find the number of comparisons for each sized data (on an average and find the best fit regression equation for the same). The mechanism for reporting shall be exactly as for LAB002.

# LAB004: Implementation of Selection sort

The aim is to implement selection sort in its standard implementation. The data to be used is the same as generated in the LAB001. Find the actual process time taken for each sized data (on an average and find the best fit regression equation for the same). Also find the number of comparisons for each sized data (on an average and find the best fit regression equation for the same). The mechanism for reporting shall be exactly as for LAB002.

# LAB005: Implementation of Shell sort

The aim is to implement shell sort in its standard implementation. The data to be used is the same as generated in the LAB001. Find the actual process time taken for each sized data (on an average and find the best fit regression equation for the same). Also find the number of comparisons for each sized data (on an average and find the best fit regression equation for the same). The mechanism for reporting shall be exactly as for LAB002.

# LAB006: Implementation of Quicksort

The aim is to implement Quicksort in its standard implementation. The data to be used is the same as generated in the LAB001. Find the actual process time taken for each sized data (on an average and find the best fit regression equation for the same). Also find the number of comparisons for each sized data (on an average and find the best fit regression equation for the same). The mechanism for reporting shall be exactly as for LAB002.

# LAB007: Implementation of Heapsort

The aim is to implement Heapsort in its standard implementation. The data to be used is the same as generated in the LAB001. Find the actual process time taken for each sized data (on an average and find the best fit regression equation for the same). Also find the number of comparisons for each sized data (on an average and find the best fit regression equation for the same). The mechanism for reporting shall be exactly as for LAB002.

# LAB008: Implementation of Mergesort

The aim is to implement two way mergesort in its standard implementation. The data to be used is the same as generated in the LAB001. Find the actual process time taken for each sized data (on an average and find the best fit regression equation for the same). Also find the number of comparisons for each sized data (on an average and find the best fit regression equation for the same). The mechanism for reporting shall be exactly as for LAB002.

# LAB009: Implementation of Randomized Quicksort

The aim is to implement Quicksort in its randomized implementation. The data to be used is the same as generated in the LAB001. Find the actual process time taken for each sized data (on an average and find the best fit regression equation for the same). Also find the number of comparisons for each sized data (on an average and find the best fit regression equation for the same). The mechanism for reporting shall be exactly as for LAB002.

# Declaration of Comparison based Sorting Techniques

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//FILE: mySort.h

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#ifndef \_\_MYSORT\_\_

#define \_\_MYSORT\_\_

//All functions return the number of comparisons

unsigned long InsertionSort(unsigned long data[], long Size);

unsigned long BubbleSort(unsigned long data[], long Size);

unsigned long SelectionSort(unsigned long data[], long Size);

unsigned long ShellSort(unsigned long data[], long Size);

unsigned long QuickSort(unsigned long data[], long Size);

unsigned long HeapSort(unsigned long data[], long Size);

unsigned long MergeSort(unsigned long data[], long Size);

unsigned long RandomizedQuickSort(unsigned long data[], long Size);

#endif // \_\_MYSORT\_\_

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Implement in **mySort.cpp**

# EXTRA001: Empirical Comparison of Sorting Algorithms (Extra Assignment)

Use the code written in LAB002 to LAB009 to empirically analyse your results. Make specific study for small N. The mechanism for reporting shall be exactly as for LAB002.

This completes the PART 001 of the experiments.

# LAB010: Implementation of Counting sort

The aim is to implement Counting sort. The data to be used is the same as generated in the LAB001. The counting behaviour is to sort in Θ(n+k). Here *n* is the number of elements to be sorted and *k* is the maximum value of the numbers in the list. For Θ(n) behaviour we require k = n. Thus the data in the files must be taken modulo size of the file (number of elements in the file. Find the actual process time taken for each sized data (on an average and find the best fit regression equation for the same). Otherwise, the mechanism for reporting shall be exactly as for LAB002.

**Describe the algorithm, establish its correctness and analyse its timing**

The output should be something like:

**N Average Time**

5 2.5e-07

10 3.3e-07

25 5.13333e-07

50 9.46667e-07

100 1.59667e-06

250 3.56667e-06

500 8.42667e-06

1000 1.353e-05

2500 3.95967e-05

5000 6.99133e-05

10000 0.000142763

25000 0.000417763

50000 0.00101024

100000 0.00232621

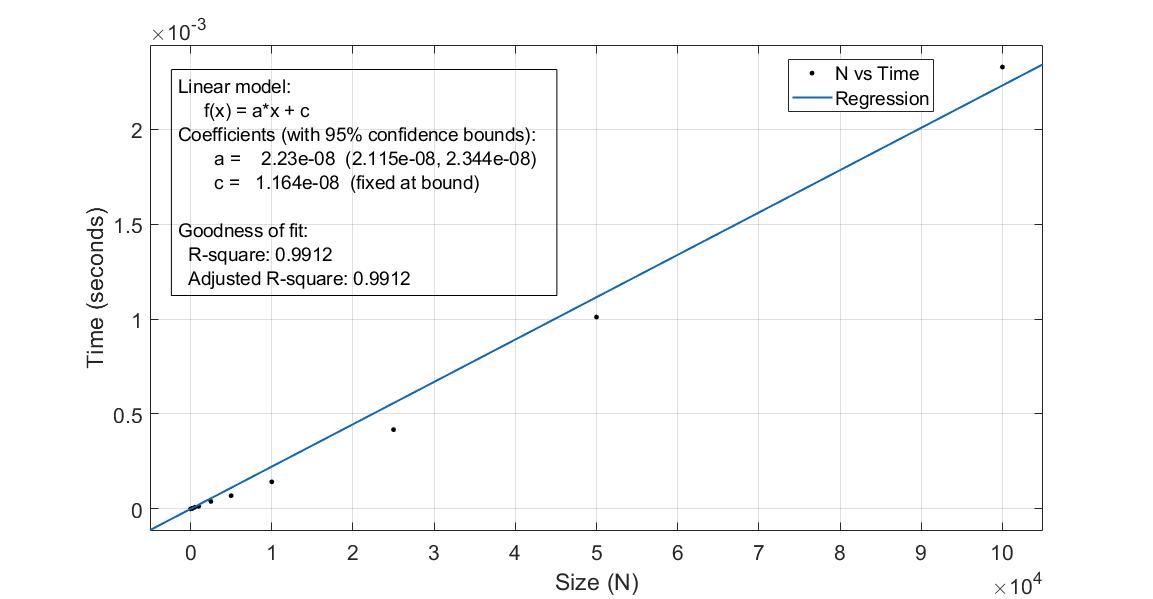


Fig 1: Plot of size of samples vs time taken.

**Summarize your findings**

**Attach your code here.**

**Note: the syntax for the implementation is after LAB0 12 description for all non-comparison based sorting experiments**

# LAB011: Implementation of Radix sort

The aim is to implement Radix sort. The data to be used is the same as generated in the LAB001. Find the actual process time taken for each sized data (on an average and find the best fit regression equation for the same). Otherwise, the mechanism for reporting shall be exactly as for LAB002.

**Describe the algorithm, establish its correctness and analyse its timing**

The output should be something like:

**N Average Time**

5 1.17e-06

10 1.57333e-06

25 2.78667e-06

50 5.44333e-06

100 1.01033e-05

250 2.23567e-05

500 4.74667e-05

1000 8.94833e-05

2500 0.000210043

5000 0.000428937

10000 0.000846957

25000 0.00244384

50000 0.00439085

100000 0.00851348

**Your regression analysis here**

**Summarize your findings**

**Attach your code here.**

**Note: the syntax for the implementation is after LAB0 12description for all non-comparison based sorting experiments**

# LAB012: Implementation of Bucket sort

The aim is to implement Bucket sort. The data to be used is the same as generated in the LAB001, however divide all numbers in the files by (long long ) 4294967295 (to get the numbers between 0 and 1) . Make the bucket sizes on the basis of number of elements to be sorted. Find the actual process time taken for each sized data (on an average and find the best fit regression equation for the same).

**Describe the algorithm, establish its correctness and analyse its timing**

Your output should be something like this (dependent on the bucket structure and size) (for doubles):

**N Average Time Average Count**

5 4.18667e-06 2.7

10 4.61333e-06 12.2333

25 7.77e-06 78.0667

50 1.05533e-05 317.467

100 2.22067e-05 846.567

250 4.07667e-05 2591.97

500 7.448e-05 5668.53

1000 0.000127487 11929.5

2500 0.00031521 30519

5000 0.000646947 61955.2

10000 0.00140012 124602

25000 0.00343644 311937

50000 0.0076977 625088

100000 0.0160014 1.24939e+06

**Your regression analysis here**

**Summarize your findings**

**Attach your code here.**

**Note: the syntax for the implementation is after LAB0 12 description for all non-comparison based sorting experiments**

# Definitions of the non-comparison based sorting techniques

The definitions are made part of the file mySort.h as:

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//FILE: mySort.h

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#ifndef \_\_MYSORT\_\_

#define \_\_MYSORT\_\_

void SwapUL(unsigned long \*a, unsigned long \*b);

unsigned long getMaxUL(unsigned long data[], long Size);

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

/\* Comparison based sorting \*/

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

//All functions return the number of comparisons

unsigned long InsertionSort(unsigned long data[], long Size);

unsigned long BubbleSort(unsigned long data[], long Size);

unsigned long SelectionSort(unsigned long data[], long Size);

unsigned long ShellSort(unsigned long data[], long Size);

unsigned long QuickSort(unsigned long data[], long Size);

unsigned long HeapSort(unsigned long data[], long Size);

unsigned long MergeSort(unsigned long data[], long Size);

unsigned long RandomizedQuickSort(unsigned long data[], long Size);

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

/\* Non-Comparison based sorting \*/

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

//Sorting in linear time

unsigned long\* CountingSort(unsigned long data[], long Size);

// The algorithm copies the auxiliary memory into data,

// return is same as data (extra time taken to copy)

unsigned long\* RadixSort(unsigned long data[], long Size);

// return is same as data (extra time taken to copy)

unsigned long BucketSort(double data[], long Size);

// sorted value returned in data. You may use the STL vector to implement

// Returns the number of comparisons. Buckets sorted using Insertion sort

// You may not implement the following (EXTRA002)

unsigned long BucketSort(unsigned long data[], long Size);

// sorted value returned in data. You may use the STL vector to implement

// Returns the number of comparisons. Buckets sorted using Insertion sort

#endif // \_\_MYSORT\_\_

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Obviously, the implementation is in **mySort.cpp**

# EXTRA002: Implementation of Bucket sort for Integers (Extra Assignment)

The aim is to implement Bucket sort for integers. The data to be used is the same as generated in the LAB001. Make the bucket sizes on the basis of number of elements to be sorted. Find the actual process time taken for each sized data (on an average and find the best fit regression equation for the same). Also find the number of comparisons for each sized data (on an average and find the best fit regression equation for the same). The mechanism for reporting shall be exactly as for LAB002.

**Describe the algorithm, establish its correctness and analyse its timing**

Your output should be something like this (dependent on the bucket structure and size):

**N Average Time Average Count**

5 4.92333e-06 4.83333

10 4.47667e-06 20.8333

25 6.38e-06 91.0667

50 9.29667e-06 247.6

100 1.52233e-05 650.267

250 3.306e-05 2011

500 7.007e-05 4826.5

1000 0.00015481 9744.6

2500 0.000340613 24002.7

5000 0.000701633 48035.2

10000 0.00142416 96282

25000 0.00338025 240460

50000 0.0075824 481058

100000 0.0137628 962541

**Your regression analysis here**

**Summarize your findings**

**Attach your code here.**

# EXTRA003: Implementation of Strassen’s Algorithm for Multiplication of Compatible Matrices (Extra Assignment)

The aim is to implement Strassen’s algorithm for multiplication of two matrices of order n times n such that n is an exact power of two first. Then implement it for multiplication of two compatible matrices of arbitrary orders.

# LAB013: Implement the Min, Max, MinMax and the Select ith smallest element algorithm

This experiment has four parts:

Part 1. Implement the finding minimum algorithm.

Part 2. Implement the finding maximum algorithm.

Part 3. Implement the algorithm to find the minimum and maximum simultaneously.

Part 4. Implement the algorithm to find the ith smallest element in an array.

The data to be used is the files generated in LAB001.

**Reporting:**

1. For Part 1 **describe the algorithm, establish its correctness and analyse its timing and** report the summary as (Average time taken per file Vs Size of file):

**N Average Time**

5 1.33333e-07

10 1.73333e-07

25 1.86667e-07

50 2.96667e-07

100 4.96667e-07

250 8.36667e-07

500 1.41333e-06

1000 2.71667e-06

2500 6.72667e-06

5000 1.26633e-05

10000 2.98967e-05

25000 7.062e-05

50000 0.00011796

100000 0.00028099

**Your regression analysis here**

**Summarize your findings**

**Attach your code here.**

2. For Part 2 **describe the algorithm, establish its correctness and analyse its timing** report the summary as (Average time taken per file Vs Size of file):

**N Average Time**

5 8.66667e-08

10 9e-08

25 1.26667e-07

50 2.2e-07

100 3.4e-07

250 7.13333e-07

500 1.66667e-06

1000 2.51667e-06

2500 6.09333e-06

5000 1.29933e-05

10000 2.73133e-05

25000 7.354e-05

50000 0.000131973

100000 0.000273597

**Your regression analysis here**

**Summarize your findings**

**Attach your code here.**

3. For Part 3 **describe the algorithm, establish its correctness and analyse its timing** report the summary as (Average time taken per file Vs Size of file and Average count of comparisons Vs File Size):

**N Average Time Average Count**

5 2.36667e-07 6

10 2.9e-07 13

25 4.2e-07 36

50 6.23333e-07 73

100 9.6e-07 148

250 2.14e-06 373

500 3.68e-06 748

1000 6.85333e-06 1498

2500 1.79533e-05 3748

5000 3.87e-05 7498

10000 6.55067e-05 14998

25000 0.000222053 37498

50000 0.00035179 74998

100000 0.000848227 149998

**Your regression analysis here**

**Summarize your findings**

**Attach your code here.**

4. For Part 3 **describe the algorithm, establish its correctness and analyse its timing** report the summary as (Average time taken per file Vs Size of file and Average count of comparisons Vs File Size) (the data is shown for the 2nd order statistic):

**N Average Time Average Count (Comparisons)**

5 2.93333e-07 5.73333

10 4.26667e-07 16.9

25 6.66667e-07 40.7

50 1.19e-06 106.9

100 1.98333e-06 196.067

250 5.92e-06 520.567

500 1.02667e-05 1190.77

1000 1.412e-05 2016.83

2500 4.075e-05 5461.33

5000 7.261e-05 9960.6

10000 0.000172277 20871.1

25000 0.000379337 48407.4

50000 0.000724497 103240

100000 0.00180895 199527

**Your regression analysis here**

**Summarize your findings**

**Attach your code here.**

The template for implementation is as:

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//FILE: myTimer.h

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#ifndef MYTIMER\_H\_INCLUDED

#define MYTIMER\_H\_INCLUDED

#include <windows.h>

class myTimer{

LARGE\_INTEGER Frequency;

LARGE\_INTEGER startTime;

LARGE\_INTEGER endTime;

double interval;

public:

myTimer() { QueryPerformanceFrequency(&Frequency); }

void StartTimer(){ QueryPerformanceCounter(&startTime); }

void EndTimer(){ QueryPerformanceCounter(&endTime); }

double GetInterval() {

return (double) (endTime.QuadPart - startTime.QuadPart) / Frequency.QuadPart;

}

};

#endif // MYTIMER\_H\_INCLUDED

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//FILE: myRandom.h

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#ifndef \_\_MYRANDOM\_H\_INCLUDED\_\_

#define \_\_MYRANDOM\_H\_INCLUDED\_\_

class myRandom{

static const unsigned long a = 1664525L, c = 1664525L;

static unsigned long x;

public:

myRandom(unsigned long s=8462817L) { x = s;}

static void setSeed(unsigned long s){ x = s;}

static unsigned long getVal() { return x = a\*x + c;}

static unsigned long getCurVal(){ return x;}

};

#endif // MYRANDOM\_H\_INCLUDED

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**Note:** the definition of the static member myRandom::x has been moved to the myRandom.cpp file.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//FILE: myRandom.cpp

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include "myRandom.h"

unsigned long myRandom::x = 8462817L;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//FILE: myAuxFunctions.h

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#ifndef MYAUXFUNCTIONS\_H\_INCLUDED

#define MYAUXFUNCTIONS\_H\_INCLUDED

#include "myRandom.h"

#include <cstdlib>

#include <iostream>

using namespace std;

//Str to unsigned long

unsigned long StrToULong(char\* x){

unsigned long val = 0;

unsigned int i= 0;

while (x[i] != '\0'){

val = val \* 10 + (x[i++] - '0');

}

return val;

}

//Swap function

template <typename T>

void Swap(T& x, T& y);

template <typename T>

void Swap(T\* x, T\* y);

template <typename T>

T Min(T A[], long Size);

template <typename T>

T Max(T A[], long Size);

template <class T>

class Pair{

private:

T x;

T y;

public:

Pair(){x = y = 0;};

Pair(T& , T&);

Pair(Pair<T>& );

T& getX();

T& getY();

Pair<T>& operator = ( Pair<T> b);

friend ostream& operator << (ostream& os, const Pair<T>& x) {

return os << "( " << x.x << ", " << x.y << " )";

}

};

template <typename T>

Pair<T> MinMax(T A[], long Size, unsigned long \*COUNT = NULL);

template <typename T>

long Partition (T d[], long low, long high, unsigned long \*COUNT=NULL);

template <typename T>

long randomPartition (T d[], long low, long high, unsigned long \*COUNT=NULL);

template <typename T>

T\* Select(T d[], long low, long high, long i, unsigned long \*COUNT=NULL);

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

/\* Implementations \*/

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

//PUT YOUR IMPLEMENTATION OF CODE HERE

#endif // MYAUXFUNCTIONS\_H\_INCLUDED

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//FILE: main.cpp

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <iostream>

#include <fstream>

#include<sstream>

#include<string>

#include <iomanip>

#include "myTimer.h"

#include "myAuxFunctions.h"

#define NUMBER\_OF\_SIZES 14

#define NUMBER\_OF\_SAMPLES 30

using namespace std;

void Analyse( double timeTaken[NUMBER\_OF\_SIZES][NUMBER\_OF\_SAMPLES],

unsigned long COUNT[NUMBER\_OF\_SIZES][NUMBER\_OF\_SAMPLES],

double AvgTime[NUMBER\_OF\_SIZES],

double AvgCount[NUMBER\_OF\_SIZES]);

void Analyse(double timeTaken[NUMBER\_OF\_SIZES][NUMBER\_OF\_SAMPLES],double AvgTime[NUMBER\_OF\_SIZES]);

int main() {

unsigned long N[] = {5,10,25, 50,100,250, 500,1000,2500, 5000,10000,25000, 50000,100000};

double timeTaken[NUMBER\_OF\_SIZES][NUMBER\_OF\_SAMPLES];

unsigned long COUNT[NUMBER\_OF\_SIZES][NUMBER\_OF\_SAMPLES];

string fileName;

myTimer T;

{

//PART 1

for (int j = 0; j < NUMBER\_OF\_SIZES; j++) {

unsigned long Data[N[j]];

for (int i = 0; i < NUMBER\_OF\_SAMPLES; i++){

ostringstream strStream;

strStream << "NUM-" << setw(2) << setfill('0') << to\_string(i+1) << "-" << setw(6) << setfill('0') << to\_string(N[j]) << ".txt";

fileName = strStream.str();

cout << fileName << endl;

ifstream INFILE(fileName);

if (!INFILE) {

cout << "Could not open file: " << fileName << endl;

return 0;

}

for (unsigned long k = 0; k < N[j]; k++) {INFILE >> Data[k];}

INFILE.close();

T.StartTimer();

Min<unsigned long>(Data,N[j]);

T.EndTimer();

timeTaken[j][i] = T.GetInterval();

}

}

double AvgTime[NUMBER\_OF\_SIZES];

Analyse(timeTaken,AvgTime);

cout << setw(6) << "N" << " " << setw(15) << "Average Time" << endl;

for (int i = 0; i < NUMBER\_OF\_SIZES; i++ ){

cout << setw(6) << N[i] << " " << setw(15) << AvgTime[i] << endl;

}

ofstream OUTFILE("OUTPUT\_MIN.txt",ios::out | ios::trunc);

if (!OUTFILE) {

cout << "Could not open file: OUTPUT\_MIN.txt" << endl;

return 0;

}

OUTFILE << setw(6) << "N" << " " << setw(15) << "Average Time" << endl;

for (int i = 0; i < NUMBER\_OF\_SIZES; i++ ){

OUTFILE << setw(6) << N[i] << " " << setw(15) << AvgTime[i] << endl;

}

OUTFILE.close();

}

{

//PART 2

for (int j = 0; j < NUMBER\_OF\_SIZES; j++) {

unsigned long Data[N[j]];

for (int i = 0; i < NUMBER\_OF\_SAMPLES; i++){

ostringstream strStream;

strStream << "NUM-" << setw(2) << setfill('0') << to\_string(i+1) << "-" << setw(6) << setfill('0') << to\_string(N[j]) << ".txt";

fileName = strStream.str();

cout << fileName << endl;

ifstream INFILE(fileName);

if (!INFILE) {

cout << "Could not open file: " << fileName << endl;

return 0;

}

for (unsigned long k = 0; k < N[j]; k++) {INFILE >> Data[k];}

INFILE.close();

T.StartTimer();

Max<unsigned long>(Data,N[j]);

T.EndTimer();

timeTaken[j][i] = T.GetInterval();

}

}

double AvgTime[NUMBER\_OF\_SIZES];

Analyse(timeTaken,AvgTime);

cout << setw(6) << "N" << " " << setw(15) << "Average Time" << endl;

for (int i = 0; i < NUMBER\_OF\_SIZES; i++ ){

cout << setw(6) << N[i] << " " << setw(15) << AvgTime[i] << endl;

}

ofstream OUTFILE("OUTPUT\_MAX.txt",ios::out | ios::trunc);

if (!OUTFILE) {

cout << "Could not open file: OUTPUT\_MAX.txt" << endl;

return 0;

}

OUTFILE << setw(6) << "N" << " " << setw(15) << "Average Time" << endl;

for (int i = 0; i < NUMBER\_OF\_SIZES; i++ ){

OUTFILE << setw(6) << N[i] << " " << setw(15) << AvgTime[i] << endl;

}

OUTFILE.close();

}

{

//PART 3

for (int j = 0; j < NUMBER\_OF\_SIZES; j++) {

unsigned long Data[N[j]];

for (int i = 0; i < NUMBER\_OF\_SAMPLES; i++){

ostringstream strStream;

strStream << "NUM-" << setw(2) << setfill('0') << to\_string(i+1) << "-" << setw(6) << setfill('0') << to\_string(N[j]) << ".txt";

fileName = strStream.str();

cout << fileName << endl;

ifstream INFILE(fileName);

if (!INFILE) {

cout << "Could not open file: " << fileName << endl;

return 0;

}

for (unsigned long k = 0; k < N[j]; k++) {INFILE >> Data[k];}

INFILE.close();

T.StartTimer();

COUNT[j][i] =0;

MinMax<unsigned long>(Data,N[j],&COUNT[j][i]);

T.EndTimer();

timeTaken[j][i] = T.GetInterval();

}

}

double AvgTime[NUMBER\_OF\_SIZES],AvgCount[NUMBER\_OF\_SIZES];

Analyse(timeTaken,COUNT,AvgTime, AvgCount);

cout << setw(6) << "N" << " " << setw(15) << "Average Time" << " " << setw(15) << "Average Count" << endl;

for (int i = 0; i < NUMBER\_OF\_SIZES; i++ ){

cout << setw(6) << N[i] << " " << setw(15) << AvgTime[i] << " " << setw(15) << AvgCount[i] << endl;

}

ofstream OUTFILE("OUTPUT\_MINMAX.txt",ios::out | ios::trunc);

if (!OUTFILE) {

cout << "Could not open file: OUTPUT.txt" << endl;

return 0;

}

OUTFILE << setw(6) << "N" << " " << setw(15) << "Average Time" << " " << setw(15) << "Average Count" << endl;

for (int i = 0; i < NUMBER\_OF\_SIZES; i++ ){

OUTFILE << setw(6) << N[i] << " " << setw(15) << AvgTime[i] << " " << setw(15) << AvgCount[i] << endl;

}

}

{

//PART 4

for (int j = 0; j < NUMBER\_OF\_SIZES; j++) {

unsigned long Data[N[j]];

for (int i = 0; i < NUMBER\_OF\_SAMPLES; i++){

ostringstream strStream;

strStream << "NUM-" << setw(2) << setfill('0') << to\_string(i+1) << "-" << setw(6) << setfill('0') << to\_string(N[j]) << ".txt";

fileName = strStream.str();

cout << fileName << endl;

ifstream INFILE(fileName);

if (!INFILE) {

cout << "Could not open file: " << fileName << endl;

return 0;

}

for (unsigned long k = 0; k < N[j]; k++) {INFILE >> Data[k];}

INFILE.close();

T.StartTimer();

COUNT[j][i] =0;

Select<unsigned long>(Data,0,N[j]-1,2,&COUNT[j][i]);

T.EndTimer();

timeTaken[j][i] = T.GetInterval();

}

}

double AvgTime[NUMBER\_OF\_SIZES],AvgCount[NUMBER\_OF\_SIZES];

Analyse(timeTaken,COUNT,AvgTime, AvgCount);

cout << setw(6) << "N" << " " << setw(15) << "Average Time" << " " << setw(15) << "Average Count" << endl;

for (int i = 0; i < NUMBER\_OF\_SIZES; i++ ){

cout << setw(6) << N[i] << " " << setw(15) << AvgTime[i] << " " << setw(15) << AvgCount[i] << endl;

}

ofstream OUTFILE("OUTPUT\_SELECT.txt",ios::out | ios::trunc);

if (!OUTFILE) {

cout << "Could not open file: OUTPUT.txt" << endl;

return 0;

}

OUTFILE << setw(6) << "N" << " " << setw(15) << "Average Time" << " " << setw(15) << "Average Count" << endl;

for (int i = 0; i < NUMBER\_OF\_SIZES; i++ ){

OUTFILE << setw(6) << N[i] << " " << setw(15) << AvgTime[i] << " " << setw(15) << AvgCount[i] << endl;

}

}

return 0;

}

void Analyse( double timeTaken[NUMBER\_OF\_SIZES][NUMBER\_OF\_SAMPLES],

unsigned long COUNT[NUMBER\_OF\_SIZES][NUMBER\_OF\_SAMPLES],

double AvgTime[NUMBER\_OF\_SIZES],

double AvgCount[NUMBER\_OF\_SIZES]){

for (int i = 0; i < NUMBER\_OF\_SIZES; i++){

AvgTime[i] = 0.0;

AvgCount[i] = 0;

for (int j = 0; j < NUMBER\_OF\_SAMPLES; j++) {

AvgTime[i] += timeTaken[i][j];

AvgCount[i] += COUNT[i][j];

}

AvgTime[i] /= double(NUMBER\_OF\_SAMPLES);

AvgCount[i] /= double(NUMBER\_OF\_SAMPLES);

}

}

void Analyse(double timeTaken[NUMBER\_OF\_SIZES][NUMBER\_OF\_SAMPLES],double AvgTime[NUMBER\_OF\_SIZES]){

for (int i = 0; i < NUMBER\_OF\_SIZES; i++){

AvgTime[i] = 0.0;

for (int j = 0; j < NUMBER\_OF\_SAMPLES; j++) {

AvgTime[i] += timeTaken[i][j];

}

AvgTime[i] /= double(NUMBER\_OF\_SAMPLES);

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

For this practical, almost the full source code is provided. You may choose to implement the code without using template but the functions should be declared in the manner we have notified for myAuxFunctions.h file.

**YOUR OUTPUT HERE**

# Lab014: Implementation of an unordered list.

The full code is being provided except the code already written previously. Including a rudimentary main function. The student should analyse each function and find out what is the time taken for each operation, establish the correctness of the functions / methods.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//FILE: myAuxFunctions.h

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//Only the function :

unsigned long StrToULong(char\* x);

//defined in this header is required for the current practical.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//FILE: myList.h

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#ifndef MYLIST\_H\_INCLUDED

#define MYLIST\_H\_INCLUDED

#include <iostream>

#include <cstdlib>

using namespace std;

template <class T>

class Head; //forward declaration

template <class T>

struct Node{

public:

Head<T> \*Parent;

Node<T> \*Next;

T Data;

Node() { Parent = NULL; Next = NULL; }

};

template <class T>

struct Head{

Node<T> \*Front;

Node<T> \*Tail;

int Size;

Head() { Front = Tail = NULL; Size = 0; }

};

template <class T>

class myList; //Forward reference

template <class T>

myList<T> operator + ( myList<T> &a, myList<T> &b); // Forward reference

template <class T>

ostream& operator << (ostream& os, const myList<T> &List); // Forward reference

template <class T>

class myList{

protected:

Head<T> H;

public:

//Constructors

explicit myList(): H() {}

explicit myList(T x);

explicit myList(Node<T>& x);

myList(const myList<T>& x);

//Destructor

virtual ~myList() { Clear(); }

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

/\* Operational functions \*/

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

Node<T>\* First();

Node<T>\* Last();

bool isEmpty();

int SizeOf();

Node<T>\* Contains(T x);

Node<T>\* Remove(T x);//only removes the node if it exists, remember to delete once used up

Node<T>\* RemoveLast(); //only removes the node if it exists, remember to delete once used up

Node<T>\* RemoveFirst(); //only removes the node if it exists, remember to delete once used up

Node<T>\* addFront(T x);

Node<T>\* addFront(Node<T>\* x);//assumes x was dynamically allocated

Node<T>\* addLast(T x);

Node<T>\* addLast(Node<T>\* x); // attaches the node x, make x from dynamic memory

Node<T>\* addAfter(T x, T y);

Node<T>\* addAfter(T x, Node<T> \*y);

void Clear(); // Resets the list

myList<T>& operator = (myList<T> x); //overload assignment

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

/\* Friend Functions \*/

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

friend myList<T> operator + <>( myList<T> &a, myList<T> &b);

//returns c = a+b

friend ostream& operator << <>(ostream& os, const myList<T> &List);

};

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

/\* Implementation \*/

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

template <class T>

myList<T>::myList(T x): H() {

Node<T> \*t = new Node<T>;

if (!t) {

cout << "1. Memory Allocation Error.\n";

exit(1);

}

t->Parent = &H;

t->Next = NULL;

t->Data = x;

H.Front = H.Tail = t;

H.Size = 1;

}

template <class T>

myList<T>::myList(Node<T>& x): H() {

Node<T> \*t = new Node<T>;

if (!t) {

cout << "2. Memory Allocation Error.\n";

exit(1);

}

t->Parent = &H;

t->Next = NULL;

t->Data = x.Data;

H.Front = H.Tail = t;

H.Size = 1;

}

template <class T>

myList<T>::myList(const myList<T>& x) {

Node<T> \*t = x.H.Front,p;

Node<T> \*f = NULL;

for (int i = 0; i < x.H.Size; i++) {

Node<T>\* y = new Node<T>();

if (!y) {

cout << "3. Memory Allocation Error.\n";

exit(1);

}

y->Data = t->Data;

y->Parent = &H;

if (H.Front == NULL) H.Front = y;

else if (i == x.H.Size - 1) {

f->Next = y;

y->Next = NULL;

H.Tail = y;

H.Size = x.H.Size;

}

else f->Next = y;

t = t->Next;

f = y;

}

}

template <class T>

Node<T>\* myList<T>::First() {

return H.Front;

}

template <class T>

Node<T>\* myList<T>::Last() {

return H.Tail;

}

template <class T>

bool myList<T>::isEmpty() {

return H.Size == 0;

}

template <class T>

int myList<T>::SizeOf(){

return H.Size;

}

template <class T>

Node<T>\* myList<T>::Contains(T x) {

Node<T> \*t = H.Front;

while (t ) {

if (t->Data == x) { return t;}

t = t->Next;

}

return NULL;

}

template <class T>

Node<T>\* myList<T>::Remove(T x) {//only removes the node if it exists, remember to delete once used up

Node<T> \*t = H.Front;

if (t->Data == x) return RemoveFirst();

while (t->Next && t->Next->Data != x) t = t->Next;

if(t->Next) {

Node<T> \*p = t->Next;

t->Next = p->Next;

H.Size--;

p->Parent = NULL;

p->Next = NULL;

return p;

}

return t->Next;

}

template <class T>

Node<T>\* myList<T>::RemoveLast() {//only removes the node if it exists, remember to delete once used up

Node<T> \*t = H.Front;

if (!t) return t; // no item present

if (H.Size == 1) RemoveFirst();

while (t->Next->Next) t = t->Next;

Node<T> \*p = t->Next;

t->Next = p->Next;

H.Tail = t ;

p->Parent = NULL;

p->Next = NULL;

H.Size--;

return p;

}

template <class T>

Node<T>\* myList<T>::RemoveFirst() {//only removes the node if it exists, remember to delete once used up

Node<T> \*t = H.Front;

if (!t) return t; // no item present

H.Front = t->Next;

H.Size--;

t->Parent = NULL;

t->Next = NULL;

return t;

}

template <class T>

Node<T>\* myList<T>::addFront(T x) {

Node<T> \*t = new Node<T>;

if (!t) {

cout << "4. Memory Allocation Error.\n";

exit(1);

}

if (H.Front == NULL) H.Tail = t;

t->Next = H.Front;

t->Parent = &H;

t->Data = x;

H.Front = t;

H.Size++;

return t;

}

template <class T>

Node<T>\* myList<T>::addFront(Node<T>\* x) {//assumes x was dynamically allocated

if (H.Front == NULL) H.Tail = x;

x->Next = H.Front;

x->Parent = &H;

H.Front = x;

H.Size++;

return x;

}

template <class T>

Node<T>\* myList<T>::addLast(T x) {

if (H.Front == NULL) return addFront(x);

Node<T> \*t = new Node<T>;

if (!t) {

cout << "5. Memory Allocation Error.\n";

exit(1);

}

t->Parent = &H;

t->Data = x;

H.Tail->Next = t;

t->Next = NULL;

H.Tail = t;

H.Size++;

return t;

}

template <class T>

Node<T>\* myList<T>::addLast(Node<T>\* x) {// attaches the node x, make x from dynamic memory

if (H.Front == NULL) return addFront(x);

x->Parent = &H;

H.Tail->Next = x;

x->Next = NULL;

H.Tail = x;

H.Size++;

return x;

}

template <class T>

Node<T>\* myList<T>::addAfter(T x, T y) {

Node<T> \*t = H.Front;

if (t == NULL) return t;

while (t->Next) {

if (t->Data == x) break;

t = t->Next;

}

if (!t->Next) return NULL;

Node<T> \*tmp = new Node<T>;

if (!tmp) {

cout << "6. Memory Allocation Error.\n";

exit(1);

}

tmp->Next = t->Next;

tmp->Parent = t->Parent;

tmp->Data = y;

t->Next = tmp;

H.Size++;

return tmp;

}

template <class T>

Node<T>\* myList<T>::addAfter(T x, Node<T> \*y) {

Node<T> \*t = H.Front;

if (t == NULL) return t;

while (t->Next) {

if (t->Data == x) break;

t = t->Next;

}

if (!t->Next) return NULL;

y->Next = t->Next;

y->Parent = t->Parent;

t->Next = y;

H.Size++;

return y;

}

template <class T>

void myList<T>::Clear() {

Node<T> \*t=H.Front,\*p;

if (!t) return;

while(t->Next) {

p = t;

t = t->Next;

delete p;

}

delete t;

H.Front = H.Tail = NULL;

H.Size = 0;

}

template <class T>

myList<T>& myList<T>::operator = (myList<T> x) {

Clear();

//cout << "X " << x;

Node<T> \*t = x.H.Front,p;

Node<T> \*f = NULL;

for (int i = 0; i < x.H.Size; i++){

Node<T>\* y = new Node<T>();

if (!y) {

cout << "7. Memory Allocation Error.\n";

exit(1);

}

y->Data = t->Data;

y->Parent = &H;

if (H.Front == NULL) H.Front = y;

else if (i == x.H.Size - 1) {

f->Next = y;

y->Next = NULL;

H.Tail = y;

H.Size = x.H.Size;

}

else f->Next = y;

t = t->Next;

f = y;

}

return \*this;

}

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

template <class T>

myList<T> operator + ( myList<T> &a, myList<T> &b){

myList<T> c; //= new myList<T>;

Node<T> \*t;

if (a.SizeOf() > b.SizeOf()) {

c = a;

t = b.H.Front;

}

else {

c = b;

t = a.H.Front;

}

//cout << "A = " << a << "B = " << b << "C = " << c;

while(t) {

c.addLast(t->Data);

t= t->Next;

}

cout << "C = " << c;

return c;

}

template <class T>

ostream& operator << (ostream& os, const myList<T> &List) {

Node<T> \*t = List.H.Front;

if (!t) return os << "{ }" << endl;

os << "{ " ;

while (t->Next) {

os << t->Data << ", ";

t = t->Next;

}

return os << t->Data << " }" << endl;

}

#endif // MYLIST\_H\_INCLUDED

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//FILE: main.cpp

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <iostream>

#include "myList.h"

using namespace std;

int main()

{

myList<int> L;

cout << L;

L.addFront(1);

L.addFront(2);

L.addFront(3);

L.addLast(4);

L.addLast(5);

cout << L;

L.addAfter(5,6);

cout << L;

L.addAfter(10,6);

cout << L;

cout << "Value Removed = " << L.RemoveFirst()->Data << endl;

cout << L;

L.addFront(3);

cout << L;

cout << "Value Removed = " << L.RemoveLast()->Data << endl;

cout << L;

L.addLast(6);

cout << L;

myList<int> K(L);

K.addLast(30);

cout << "K = " << K;

myList<int> R;

R = K;

cout << "R = " << R;

myList<int> A;

A = K + R;

cout << "A = " << A;

return 0;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**YOUR OUTPUT HERE**

# Lab015: Implementation of a data structure and algorithms for Sets.

Implement a data structure for sets and the operations on the same as. This uses the myList.h of LAB014. The implementation shall require the following files:

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

/\* FILE: mySet.h \*/

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

#ifndef MYSET\_H\_INCLUDED

#define MYSET\_H\_INCLUDED

#include "myList.h"

#include <sstream>

template <class T>

class mySet; // Forward reference

template <class T>

mySet<T> operator + ( mySet<T> &a, mySet<T> &b); // Forward reference

template <class T>

mySet<T> operator - ( mySet<T> &a, mySet<T> &b); // Forward reference

template <class T>

mySet<T> operator \* ( mySet<T> &a, mySet<T> &b); // Forward reference

template <class T>

class mySet: public myList<T> {

private:

//myList functions not accessible to instances of mySet

public:

mySet():myList<T>() {}

mySet(T x):myList<T>(x) {}

mySet(const T\* x, long Size);

mySet(mySet<T>& x):myList<T>(x) {}

~mySet(){Clear();}

using myList<T>::First;

using myList<T>::Last;

using myList<T>::isEmpty;

using myList<T>::SizeOf;

using myList<T>::Contains;

using myList<T>::Clear;

using myList<T>::Remove;

Node<T>\* addLast(T x);

string Name() {

if (myList<T>::H.Size == 0) return "PHI";

else{

ostringstream ss;

ss << myList<T>::H.Front->Data;

return ss.str();

}

}

friend mySet<T> operator + <>( mySet<T> &a, mySet<T> &b);

mySet<T> operator += (T x);

mySet<T> operator += (mySet<T> x);

mySet<T> operator -= (T x);

mySet<T> operator -= (mySet<T> x);

friend mySet<T> operator - <>( mySet<T> &a, mySet<T> &b);

mySet<T> operator \*= (mySet<T> x);

friend mySet<T> operator \* <>( mySet<T> &a, mySet<T> &b);

};

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

/\* Implementation \*/

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

template <class T>

mySet<T>::mySet(const T\* x, long Size):myList<T>(){

for (long i = 0; i< Size; i++) addLast(x[i]);

}

template <class T>

Node<T>\* mySet<T>::addLast(T x) {

if (Contains(x)) return NULL;

Node<T> \*t = new Node<T>;

if (!t) {

cout << "5. Memory Allocation Error.\n";

exit(1);

}

if (myList<T>::H.Front == NULL) myList<T>::H.Front = t;

else myList<T>::H.Tail->Next = t;

t->Parent = &(myList<T>::H);

t->Data = x;

t->Next = NULL;

myList<T>::H.Tail = t;

myList<T>::H.Size++;

return t;

}

template <class T>

mySet<T> mySet<T>::operator += (T x) {

addLast(x);

return \*this;

}

template <class T>

mySet<T> mySet<T>::operator += (mySet<T> x){

Node<T> \*t = myList<T>::H.Front;

while (t) {

addLast(t->Data);

t = t->Next;

}

return \*this;

}

template <class T>

mySet<T> mySet<T>::operator -= (T x){

if (Contains(x)) Remove(x);

return \*this;

}

template <class T>

mySet<T> mySet<T>::operator -= (mySet<T> x){

Node<T> \*t = x.H.Front;

while (t) {

if (Contains(t->Data)) Remove(t->Data);

t = t->Next;

}

return \*this;

}

template <class T>

mySet<T> mySet<T>::operator \*= (mySet<T> x){

mySet<T> c(\*this);

Node<T> \*t = c.H.Front;

while(t) {

if (!x.Contains(t->Data)) Remove(t->Data);

t = t->Next;

}

return \*this;

}

/\* FRIENDS \*/

template <class T>

mySet<T> operator - ( mySet<T> &a, mySet<T> &b){

mySet<T> c(a);

c -= b;

return c;

}

template <class T>

mySet<T> operator + ( mySet<T> &a, mySet<T> &b){

mySet<T> c;

Node<T> \*t;

if (a.SizeOf() > b.SizeOf()) {

c = a;

t = b.H.Front;

}

else {

c = b;

t = a.H.Front;

}

while(t) {

if (!a.Contains(t->Data)) c.addLast(t->Data);

t= t->Next;

}

return c;

}

template <class T>

mySet<T> operator \* ( mySet<T> &a, mySet<T> &b){

mySet<T> c,d;

Node<T> \*t;

if (a.SizeOf() > b.SizeOf()) {

c = a;

t = b.H.Front;

}

else {

c = b;

t = a.H.Front;

}

while(t) {

if (a.Contains(t->Data)) d.addLast(t->Data);

t= t->Next;

}

return d;

}

#endif // MYSET\_H\_INCLUDED

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

And the driver file for showcasing the implementation:

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

/\* FILE: main.cpp \*/

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

#include <iostream>

#include <sstream>

#include <string>

#include "mySet.h"

using namespace std;

int main() {

{

mySet<int> L;

cout << "1. L= " << L;

L.addLast(1);

cout << "2. L= " << L;

L.addLast(2);

L.addLast(3);

L.addLast(4);

L.addLast(5);

cout << "3. L= " << L;

//L.addLast(5);

//cout << "4. L= " << L;

//L += 5;

//cout << "5. L= " << L;

L+=6;

cout << "6. L= " << L;

mySet<int> R(-1);

cout << "7. R=" << R;

mySet<int> S(L);

cout << "8. S=" << S;

S -= 1;

cout << "9. S=" << S;

R -= S;

cout << "10. L=" << R;

}

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

{

int a[] = {1,2,3,4}, b[] = {2,3,4,5};

mySet<int> L(a,4),R(b,4);

cout << "11. L=" << L;

cout << "12. R=" << R;

L \*= R;

cout << "13. L=" << L;

mySet<int> S;

S = L \* R;

cout << "14. S=" << S;

cout << "15. S=" << S.Name() << endl;

S -= 2;

cout << "16. S=" << S;

S += 2;

cout << "17. S=" << S;

cout << "18. S=" << S.Name()<<endl;

}

return 0;

}

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

**YOUR OUTPUT HERE**

# Lab016: Implementation of a data structure and algorithms for Disjoint Sets.

This is a simplistic structure, where it allows us to represent Disjoint sets and do finite set theoretical operations on the sets.

The implementation is through the file:

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

/\* File: myDisjointSet.h \*/

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

#ifndef MYDISJOINTSET\_H\_INCLUDED

#define MYDISJOINTSET\_H\_INCLUDED

#include <iostream>

#include <cstdlib>

#include <sstream>

using namespace std;

template <class T>

class Head; //forward declaration

template <class T>

struct Node{

public:

Head<T> \*Parent;

Node<T> \*Next;

T Data;

Node() { Parent = NULL; Next = NULL; }

};

template <class T>

struct Head{

Node<T> \*Front;

Node<T> \*Tail;

int Size;

Head() { Front = Tail = NULL; Size = 0; }

};

template <class T>

class myDS; // Forward reference

template <class T>

myDS<T> operator + ( myDS<T> &a, myDS<T> &b); // Forward reference

template <class T>

myDS<T> operator - ( myDS<T> &a, myDS<T> &b); // Forward reference

template <class T>

myDS<T> operator \* ( myDS<T> &a, myDS<T> &b); // Forward reference

template <class T>

ostream& operator << (ostream& os, const myDS<T> &S);

template <class T>

class myDS{

private:

Head<T> H;

public:

myDS(): H() {}

myDS(T x);

myDS(Node<T>& x);

myDS(const T\* x, long Size);

myDS(const myDS<T>& x);

//Destructor

virtual ~myDS() { Clear(); }

T\* First();

T\* Last();

bool isEmpty();

int SizeOf();

bool Contains(T x);

friend ostream& operator << <> (ostream& os, const myDS<T> &S);

friend myDS<T> operator + <>(myDS<T> &a, myDS<T> &b);

friend myDS<T> operator - <>(myDS<T> &a, myDS<T> &b);

friend myDS<T> operator \* <>(myDS<T> &a, myDS<T> &b);

myDS<T> operator += (T x);

myDS<T> operator += (myDS<T> x);

myDS<T> operator -= (T x);

myDS<T> operator -= (myDS<T> x);

myDS<T> operator \*= (myDS<T> x);

string Name();

void Clear();

};

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

/\* Implementation \*/

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

template <class T>

myDS<T>::myDS(T x): H(){

Node<T> \*t = new Node<T>;

if (!t) {

cout << "1. Memory Allocation Error.\n";

exit(1);

}

t->Parent = &H;

t->Next = NULL;

t->Data = x;

H.Front = H.Tail = t;

H.Size = 1;

}

template <class T>

myDS<T>::myDS(Node<T>& x): H() {

Node<T> \*t = new Node<T>;

if (!t) {

cout << "2. Memory Allocation Error.\n";

exit(1);

}

t->Parent = &H;

t->Next = NULL;

t->Data = x.Data;

H.Front = H.Tail = t;

H.Size = 1;

}

template <class T>

myDS<T>::myDS(const T\* x, long Size):H(){

for (long i = 0; i< Size; i++) \*this +=x[i];

}

template <class T>

myDS<T>::myDS(const myDS<T>& x) {

Node<T> \*t = x.H.Front,p;

Node<T> \*f = NULL;

for (int i = 0; i < x.H.Size; i++) {

Node<T>\* y = new Node<T>();

if (!y) {

cout << "3. Memory Allocation Error.\n";

exit(1);

}

y->Data = t->Data;

y->Parent = &H;

if (H.Front == NULL) H.Front = y;

else if (i == x.H.Size - 1) {

f->Next = y;

y->Next = NULL;

H.Tail = y;

H.Size = x.H.Size;

}

else f->Next = y;

t = t->Next;

f = y;

}

}

template <class T>

T\* myDS<T>::First() {

if (H.Front) return &(H.Front->Data);

return NULL;

}

template <class T>

T\* myDS<T>::Last() {

if (H.Size != 0) return &(H.Tail->Data);

else return NULL;

}

template <class T>

bool myDS<T>::isEmpty() {

return H.Size == 0;

}

template <class T>

int myDS<T>::SizeOf(){

return H.Size;

}

template <class T>

bool myDS<T>::Contains(T x) {

Node<T> \*t = H.Front;

while (t ) {

if (t->Data == x) { return true;}

t = t->Next;

}

return false;

}

template <class T>

ostream& operator << (ostream& os, const myDS<T> &S) {

Node<T> \*t = S.H.Front;

if (!t) return os << "{ }";;

os << "{ " ;

while (t->Next) {

os << t->Data << ", ";

t = t->Next;

}

return os << t->Data << " }";

}

template <class T>

myDS<T> operator + (myDS<T> &a, myDS<T> &b){

//Remember the sets are disjoint, no check required

myDS<T> c;

Node<T> \*t;

if (a.SizeOf() > b.SizeOf()) {

c = a;

t = b.H.Front;

}

else {

c = b;

t = a.H.Front;

}

while(t) {

//if (!a.Contains(t->Data))

c += (t->Data);

t= t->Next;

}

return c;

}

template <class T>

myDS<T> operator - (myDS<T> &a, myDS<T> &b){

return a;

}

template <class T>

myDS<T> operator \* (myDS<T> &a, myDS<T> &b){

myDS<T> c;

return c;

}

template <class T>

myDS<T> myDS<T>::operator += (T x) {

//if (Contains(x)) return NULL;

Node<T> \*t = new Node<T>;

if (!t) {

cout << "5. Memory Allocation Error.\n";

exit(1);

}

if (H.Front == NULL) H.Front = t;

else H.Tail->Next = t;

t->Parent = &H;

t->Data = x;

t->Next = NULL;

H.Tail = t;

H.Size++;

return \*this;

}

template <class T>

myDS<T> myDS<T>::operator += (myDS<T> x){

Node<T> \*t = H.Front;

while (t) {

\*this += t->Data;

t = t->Next;

}

return \*this;

}

template <class T>

myDS<T> myDS<T>::operator -= (T x){

if (Contains(x)) Remove(x);

return \*this;

}

template <class T>

myDS<T> myDS<T>::operator -= (myDS<T> x){

Node<T> \*t = x.H.Front;

while (t) {

if (Contains(t->Data)) Remove(t->Data);

t = t->Next;

}

return \*this;

}

template <class T>

myDS<T> myDS<T>::operator \*= (myDS<T> x){

myDS<T> c;

return c;

}

template <class T>

string myDS<T>::Name() {

if (H.Size == 0) return "PHI";

else{

ostringstream ss;

ss << H.Front->Data;

return ss.str();

}

}

template <class T>

void myDS<T>::Clear() {

Node<T> \*t=H.Front,\*p;

if (!t) return;

while(t->Next) {

p = t;

t = t->Next;

delete p;

}

delete t;

H.Front = H.Tail = NULL;

H.Size = 0;

}

#endif // MYDISJOINTSET\_H\_INCLUDED

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

The main function may be written as:

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

/\* File: main.cpp \*/

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

#include <iostream>

#include <sstream>

#include <string>

#include "myDisjointSet.h"

using namespace std;

int main() {

char a[] = {'A', 'B', 'C', 'D'};

myDS<char> L;

cout << "1. L= " << L << endl;

L += 'U';

cout << "2. L= " << L << endl;

L += 'V';

L += 'W';

cout << "3. L= " << L << endl;

L += 'X';

cout << "4. L= " << L << endl;

myDS<char> R(a,4);

cout << "5. R= " << R << endl;

myDS<char> A;

cout << "6. A= " << (A = R+L) << endl;

cout << "7. 'a' in Set: " << A.Contains('a') << endl;

cout << "8. 'A' in Set: " << A.Contains('A') << endl;

return 0;

}

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

**YOUR OUTPUT HERE**

# Lab017: Implementation of Disjoint Set Forest (data structure and algorithms).

Implement the Disjoint Set Forest structure with path compression and Union-by-rank and Union-by-size. The classes to be implemented are in the file:

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

// FILE: myDSF.h

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

#ifndef MYDSF\_H\_INCLUDED

#define MYDSF\_H\_INCLUDED

#include <unordered\_map>

#include <vector>

#include <iostream>

template <class T>

struct Node; //forward reference

template <class T>

std::ostream& operator << (std::ostream& os, const Node<T>& x); //forward reference

template <class T>

bool operator == (const Node<T>& x, const Node<T>& y); //forward reference

template <class T>

struct Node{

T Data;

int Parent; // The index of the parent node in the DSF vector

int idx; //The index where it shall be stored by the DSF vector

int Value; //to store rank or size

Node():Data((T)(NULL)), Parent(-1), idx(-1), Value(-1){}

Node(const T x): Data(x), Parent(-1), idx(-1), Value(-1){}

explicit Node(T x, int Index): Data(x), Parent(-1), idx(Index), Value(-1){}

Node(const Node<T>& x): Data(x.Data), Parent(x.Parent), idx(x.idx), Value(x.Value){}

friend std::ostream& operator << <>(std::ostream& os, const Node<T>& x);

friend bool operator == <>(const Node<T>& x, const Node<T>& y);

Node<T>& operator = (const Node<T>& y);

};

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

/\* IMPLEMENTATION NODE STRUCTURE \*/

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

template <class T>

Node<T>& Node<T>::operator = (const Node<T>& y){

Data = y.Data;

Parent = y.Parent;

idx = y.idx;

Value = y.Value;

return \*this;

}

template <class T>

bool operator == (const Node<T>& x, const Node<T>& y){

if (x.Parent == -1) return false;

if (y.Parent == -1) return false;

return x.Parent == y.Parent;

}

template <class T>

std::ostream& operator << (std::ostream& os, const Node<T>& x){

return os << "[" << x.Data << ", " << x.Parent << ", " << x.idx << ", " << x.Value << "]";

}

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

typedef enum myDSF\_MODE{ BYSIZE, BYRANK} myDSF\_MODE;

template <class T>

class myDSF; //Forward reference

template <class T>

std::ostream& operator << (std::ostream& os, myDSF<T>& ds); //forward reference

template <class T>

class myDSF{

/\* The key is the data of the element.While the value is the index

at which the element shall be stored in the vector \*/

std::unordered\_map<T,int> myMap;

/\* The elements are stored here indexed by value of the map \*/

std::vector<Node<T>> myForest;

myDSF\_MODE mode;

public:

explicit myDSF(myDSF\_MODE m=BYSIZE): myMap(), myForest(), mode(m){}; //Make forest empty

explicit myDSF(T x, int Index,myDSF\_MODE m=BYSIZE) : myMap(), myForest(), mode(m) { }; //make forest with one element

/\* Returns true if inserted as a singleton set

false if already exists \*/

bool operator += (Node<T> x); //Implements Make Set

/\* Index of where the parent is stored, -1 if not stored \*/

int myParent(Node<T> x);

int myParent(const T x);

/\* Returns -1 if the node is not a member else returns

the idx to the representative element \*/

int mySet(const Node<T>& x); //Implements Find Set with path compression

int mySet(T x); //Implements Find Set with path compression

/\* returns true if the union was performed and false if it

could not. This can happen if one of the elements was not in

the forest \*/

int Union( Node<T> x, const Node<T> y);

friend std::ostream& operator << <> (std::ostream& os, myDSF<T>& ds);

Node<T> operator [] (int i) { //does no range checking

return myForest[i];

}

};/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

/\* IMPLEMENTATION myDSF CLASS \*/

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

//YOUR IMPLEMENTATUION CODE HERE

#endif // MYDSF\_H\_INCLUDED

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

The main function to be used for demonstrating the working is of the type:

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

// FILE: main.cpp

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

#include <iostream>

#include "myDSF.h"

using namespace std;

int main() {

Node<int> x;

Node<int> y(1);

cout << "1. X = " << x << " Y = " << y << endl;

cout << "2. X == Y" << (x == y) << endl;

cout << "-----------------\n";

Node<char> a('A',0);

myDSF<char> z(BYRANK);

z += a;

cout << "3. Z = " << z << endl;

a.idx = 1;

a.Data = 'B';

z+= a;

cout << "4. Z = " << z << endl;

for (char i = 'C'; i <= 'H'; i++) {

a.idx = (int) (i - 'A');

a.Data = i;

z+= a;

}

cout << "5. Z = " << z << endl;

z.Union('E','F'); //union 4 5

cout << "6. Z = " << z << endl;

z.Union('G','H'); //union 6 7

cout << "7. Z = " << z << endl;

z.Union('F','H'); //union 5 7

cout << "8. Z = " << z << endl;

cout << "9. Z = " << z[z.mySet('A')] << endl;

cout << "----------\n";

z.Union('Q','H');

cout << "10. Z = " << z << endl;

return 0;

}

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

**Your OUTPUT HERE**

# Lab018: Implementation of a Matrix class.

The requirement is to implement a matrix class that should allow one to:

1. Declare a matrix of specific datatype and dimensions.
2. Allow printing of matrix elements.
3. Allow addition subtraction and multiplication of matrices.
4. The concept of operator overloading may be used to implement these operations.
5. Implement a function to randomly initialize a matrix.

The implementation shall use the following files:

1. myException.h
2. myRandom.h
3. myRandom.cpp
4. The rest of the files are below:

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

/\* FILE: myArray.h

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

#ifndef MYARRAY\_H\_INCLUDED

#define MYARRAY\_H\_INCLUDED

#include <ostream>

#include <string>

#include "myException.h"

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

/\* For proper usage these class typename T should be one of the numeric data types \*/

/\* Usage with other datatypes is fraught with inconvenience. For non-numeric datatype \*/

/\* Use AT YOUR OWN RISK. \*/

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

//Forward declarations

template <class T>

class myArray;

template <class T>

class myArray {

T \*D; //Data

int S; //Size

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

/\* A private function for allocation of memory \*/

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

void Allocate(int N) {

D = new T[S=N];

if (!D) throw myException("myArray:Memory initialization failed\n");

}

public:

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

/\* CONSTRUCTOR(S) \*/

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

myArray<T>(){ D = NULL; S = -1;} //noninitialized array

myArray<T>(unsigned int Size) {

Allocate(int(Size));

for (int i = 0; i < S; i++) D[i] =T(0);

}

myArray<T>(T \*d, unsigned int Size) {

Allocate(int(Size));

for (int i = 0; i < S; i++) D[i] =d[i];

}

myArray<T>(const myArray<T>& A) {

Allocate(A.S);

for (int i = 0; i < S; i++) D[i] =A.D[i];

}

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

/\* DESTRUCTOR \*/

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

virtual ~myArray(){ Clear(); }

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

/\* Auxiliary functions but useful \*/

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

bool isEmpty() { return S == -1;} // If data exists false else true

int Size() {if (S < 0) return 0; return S;} // Number of data items put in array

void Clear(){ delete [] D; S = -1; D = NULL;} // Truncates the array

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

/\* OPERATOR OVERLOADING \*/

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

T& operator[] (int i){ //Access by index (zero based)

if (i > -1 && i < S) return D[i];

throw myException("myArray: Out of range access\n");

}

myArray<T>& operator = (myArray<T> x){ //asignment

if (x.Size() <= 0) throw myException("myArray: Assigning non initialized array\n");

Clear();

Allocate(x.S);

for (int i = 0; i < S; i++) D[i] = x.D[i];

return \*this;

}

myArray<T>& operator - () { //unary -

if (Size() <= 0) throw myException("myArray: negative of unassigned array\n");

myArray<T> x(\*this);

for (int i = 0; i < S; i++) x[i] = -x[i];

return x;

}

myArray<T>& operator + () {//unary +

if (Size() <= 0) throw myException("myArray: negative of unassigned array\n");

myArray<T> x(\*this);

return x;

}

myArray<T>& operator += (myArray<T> x){ //add + assignement

if (x.Size() != Size()) throw myException("myArray +: Different sized array\n");

for (int i = 0; i < x.Size(); i++) D[i] += x[i];

return \*this;

}

friend myArray<T> operator + (myArray<T> x, myArray<T> y){ //addition

if (x.Size() != y.Size()) throw myException("myArray +: Different sized array\n");

myArray<T> z(x.Size());

for (int i = 0; i < x.Size(); i++) z[i] = x[i] + y[i];

return z;

}

myArray<T>& operator -= (myArray<T> x){ //subtract and assign

if (x.Size() != Size()) throw myException("myArray +: Different sized array\n");

for (int i = 0; i < x.Size(); i++) D[i] -= x[i];

return \*this;

}

friend myArray<T> operator - (myArray<T> x, myArray<T> y){ //subtract

if (x.Size() != y.Size()) throw myException("myArray +: Different sized array\n");

myArray<T> z(x.Size());

for (int i = 0; i < x.Size(); i++) z[i] = x[i] - y[i];

return z;

}

friend T operator % (myArray<T> x, myArray<T> y){ //dot product

if (x.Size() != y.Size()) throw myException("myArray +: Different sized array\n");

T z= T(0);

for (int i = 0; i < x.Size(); i++) z += x[i] \* y[i];

return z;

}

friend myArray<T> operator \* (myArray<T> a, T x){ //multiply by a constant

if (a.Size() <= 0) throw myException("myArray: negative of unassigned array\n");

myArray<T> y(a);

for (int i = 0; i < y.Size(); i++) y[i] = a[i] \* x;

return y;

}

friend myArray<T> operator \* (T x, myArray<T> a){ //multiply by a constant

return a \* x;

}

//OUTPUT STREAM OVERLOAD. NOT A VERY TRICKY FUNCTION

friend std::ostream& operator << (std::ostream& os, myArray<T> x){

if (x.Size() <= 0) return os << "{ Null }";

for (int i = 0; i < x.Size() - 1; i++ ) os << x[i] << " ";

return os << x[x.Size() - 1];

}

//OUTPUT in a slighly formatted manner

void Print(std::ostream& os=std::cout ){

if (Size() <= 0) {os << "{ Null }"; return;}

os << "{ ";

for (int i = 0; i < Size()-1; i++ ) os << D[i] << " ";

os << D[Size() - 1] << " }";

}

// Read an array. The size of the array is fixed before reading

friend std::istream& operator >> (std::istream& Input, myArray<T>& x){

int n=x.S;

for (int i = 0; i< n; i++) Input >> x[i];

return Input;

}

};

#endif // MYARRAY\_H\_INCLUDED

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

/\* FILE: Matrix.h

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

#ifndef MATRIX\_H\_INCLUDED

#define MATRIX\_H\_INCLUDED

#include "myRandom.h"

#include "myArray.h"

// Implementing a 2D array as an array of arrays

// Try implement them as a linear array, keep track of indices

// Note:

// A matrix has M rows and N Columns

// For purpose of entry and output, by overloading the sequence of printing shall

// ROW(M) COLS(N) A11 A12 A13 ... A1N A21 A22 A23 ... A2N ... AM1 AM2 AM3 ... AMN

// For pretty printing Print method is provided

template <class T>

class Matrix{

myArray<T> \*D; //Data

int M,N; //Size

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

/\* A private function for allocation of memory \*/

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

void Allocate(int m, int n) {

M = m;

N = n;

D = new myArray<T>[M];

for (int i = 0; i < M; i++) D[i] = myArray<T>(N);

if (!D) throw myException("Matrix:Memory initialization failed\n");

}

public:

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

/\* CONSTRUCTOR(S) \*/

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

Matrix<T>(){ D = NULL; M = -1;} //noninitialized array

Matrix<T>(unsigned int m, unsigned n){ Allocate(int(m),int(n)); }

Matrix<T>(T \*d,unsigned int m, unsigned n):M(m), N(n) {

Allocate(int(m),int(n));

for (int i = 0; i < M; i++)

for (int j = 0; j < N; j++)

D[i][j] = \*((d+i\*n) + j);

}

Matrix<T>(const Matrix<T>& A) {

D = new myArray<T>[M = A.M];

N = A.N;

if (!D) throw myException("Matrix: Memory initialization failed\n");

for (int i = 0; i < M; i++) D[i] = A.D[i];

}

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

/\* DESTRUCTOR \*/

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

~Matrix(){ Clear(); }

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

/\* Auxiliary functions but useful \*/

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

bool isEmpty() { return M == -1;}

int Rows() {if (M < 0) return 0; return M;}

int Cols() {if (M < 0) return 0; return D[0].Size();}

void Clear(){ delete [] D; M = -1; D = NULL;}

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

/\* OPERATOR OVERLOADING \*/

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

myArray<T>& operator[] (int i){ //Access by index

if (i > -1 && i < M) return D[i];

throw myException("Matrix: Out of range access\n");

}

friend std::ostream& operator << (std::ostream& os, Matrix<T> x) { // OUTPUT, not very well formatted

if (x.M<0) return os << "0 0";

else os << x.M << " " << x.N << " ";

for (int i = 0; i < x.M ; i++) os << x.D[i] << " ";

return os;// << x.D[x.M - 1];

}

void Print(std::ostream& os=std::cout) { // OUTPUT, slightly formatted

if (M<0) {os << "{ NULL }"; return;}

else os << "{ " << M << ", " << N << " }";

for (int i = 0; i < M; i++) D[i].Print(os);

}

Matrix<T>& operator = (Matrix<T> x){ //asignment

if (x.M <= 0 || x.N <= 0) throw myException("Matrix 01: Assigning non initialized matrix\n");

Clear();

Allocate(x.M,x.N);

for (int i = 0; i < M; i++) D[i] = x.D[i];

return \*this;

}

Matrix<T>& operator - () { //unary -

if (M <= 0 || N <= 0) throw myException("Matrix 02: non initialized matrix\n");

Matrix<T> x(\*this);

for (int i = 0; i < M; i++) x.D[i] = -D[i];

return x;

}

Matrix<T>& operator + () {//unary +

if (M <= 0 || N <= 0) throw myException("Matrix 03: non initialized matrix\n");

myArray<T> x(\*this);

return x;

}

Matrix<T>& operator += (Matrix<T> x){ //add + assignment

if (x.M !=M || x.N != N) throw myException("Matrix 04: Adding incompatible matrix\n");

for (int i = 0; i < x.M; i++) D[i] += x[i];

return \*this;

}

friend Matrix<T> operator + (Matrix<T> x, Matrix<T> y){ //addition

if (x.M !=y.M || x.N != y.N) throw myException("Matrix 05: Adding incompatible matrix\n");

Matrix<T> z(x);

for (int i = 0; i < x.M; i++) z.D[i] = x.D[i] + y.D[i];

return z;

}

Matrix<T>& operator -= (Matrix<T> x){ //subtract and assign

if (x.M !=M || x.N != N) throw myException("Matrix 06: Subtracting incompatible matrix\n");

for (int i = 0; i < x.M; i++) D[i] -= x.D[i];

return \*this;

}

friend Matrix<T> operator - (Matrix<T> x, Matrix<T> y){ //subtract

if (x.M !=y.M || x.N != y.N) throw myException("Matrix 07: Subtracting incompatible matrix\n");

Matrix<T> z(x);

for (int i = 0; i < x.M; i++) z.D[i] = x.D[i] - y.D[i];

return z;

}

friend Matrix<T> operator % (Matrix<T> x, Matrix<T> y){ //Term \* Term multiplication

if (x.M !=y.M || x.N != y.N) throw myException("Matrix 08: Incompatible matrix\n");

Matrix<T> z(x);

for (int i = 0; i < x.M; i++)

for (int j = 0; j < x.N; j++)

z[i][j] = x[i][j] \* y[i][j];

return z;

}

friend Matrix<T> operator \* (Matrix<T> a, T x){ //multiply by a scaler

if (a.M <= 0 || a.N <= 0) throw myException("Matrix 09: non initialized matrix\n");

Matrix<T> y(a);

for (int i = 0; i < y.M; i++) y.D[i] = a.D[i] \* x;

return y;

}

friend Matrix<T> operator \* (T x, Matrix<T> a){ //multiply by a scaler

return a \* x;

}

friend Matrix<T> operator \* (Matrix<T> a, Matrix<T> b){//multiply

if (a.N != b.M ) throw myException("Matrix 11: Incompatible matrix\n");

Matrix<T> y(a.M,b.N);

for (int i = 0; i < y.M; i++)

for (int j = 0; j < y.N; j++)

for (int k = 0; k < a.N; k++) y[i][j] += a[i][k] \* b[k][j];

return y;

}

Matrix<T>& operator \*= (Matrix<T> b){//multiply and assign

if (N != b.M ) throw myException("Matrix 10: Incompatible matrix\n");

Matrix<T> y(M,b.N);

for (int i = 0; i < y.M; i++)

for (int j = 0; j < y.N; j++)

for (int k = 0; k < N; k++) y[i][j] += D[i][k] \* b[k][j];

\*this = y;

return \*this;

}

Matrix<T> operator ~ (){//Transpose

if (M <= 0 || N <= 0) return Matrix<T>();

Matrix<T> y(N,M);

for (int i = 0; i < M; i++)

for (int j = 0; j < N; j++) y[j][i] += D[i][j];

return y;

}

friend std::istream& operator >> (std::istream& Input, Matrix<T>& x){ //read a matrix, first 2 elements are the size parameters ROWS COLS

x.Clear();

int m,n;

Input >> m >> n;

x.Allocate(m,n);

for (int i = 0; i< m; i++) Input >> x[i];

return Input;

}

};

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

/\* A SIMPLE FUNCTIONS TO SHOWCASE SOME USAGE \*/

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

#include <type\_traits>

myRandom R;

template <typename T>

Matrix<T> RandomMatrix(const int M, const int N, long MAX = 0){

Matrix<T> x(M,N);

unsigned long V;

for (int i = 0; i < M; i++){

for (int j = 0; j < N; j++) {

V = R.getVal();

if (std::is\_same<T,double>::value || std::is\_same<T,float>::value) x[i][j] = (V / double(4294967295LL));

else if (std::is\_same<T,int>::value) x[i][j] = V - long(V/INT\_MAX)\*INT\_MAX;

else if (std::is\_same<T,long>::value) x[i][j] = V - long(V/LONG\_MAX)\*LONG\_MAX;

if (MAX) {

if (std::is\_same<T,double>::value || std::is\_same<T,float>::value) x[i][j] \*= MAX;

else x[i][j] -= long(x[i][j]/MAX)\*MAX;

}

}

}

return x;

}

#endif // MATRIX\_H\_INCLUDED/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

A sample main.cpp is gen. You should write your own to demonstrate the shortcomings of the code.

Replace this main.cpp by your own and the output by your program output.

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

/\* FILE: main.cpp

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

#include <iostream>

#include <fstream>

#include "Matrix.h"

using namespace std;

int main() {

//Creates a set of matrices

Matrix<int> A, B, C;

Matrix<long> D, E, F;

Matrix<double> G, H, I;

//INITIALIZE BY INVOKING RANDOMMATRIX

A = RandomMatrix<int>(3,3,2);

B = RandomMatrix<int>(3,3,5);

C = RandomMatrix<int>(3,3,10);

D = RandomMatrix<long>(3,3,4);

E = RandomMatrix<long>(3,3,6);

F = RandomMatrix<long>(3,3,8);

G = RandomMatrix<double>(3,3,4);

H = RandomMatrix<double>(3,3,1);

I = RandomMatrix<double>(3,3,2);

//PRINT THEM OUT

cout << "A = "; A.Print(); cout << endl;

cout << "B = "; B.Print(); cout << endl;

cout << "C = "; C.Print(); cout << endl;

//PRINT THEM OUT

cout << "D = "; F.Print(); cout << endl;

cout << "E = "; E.Print(); cout << endl;

cout << "F = "; D.Print(); cout << endl;

//PRINT THEM OUT

cout << "G = "; G.Print(); cout << endl;

cout << "H = "; H.Print(); cout << endl;

cout << "I = "; I.Print(); cout << endl;

//OPERATIONS ON SAME TYPE

//PRINT THEM OUT

cout << "A + B = "; (A+B).Print(); cout << endl;

cout << "A - B = "; (A-B).Print(); cout << endl;

cout << "A \* B = "; (A\*B).Print(); cout << endl;

//PRINT THEM OUT

cout << "D + F = "; (D+F).Print(); cout << endl;

cout << "D - F = "; (D-F).Print(); cout << endl;

cout << "D \* F = "; (D\*F).Print(); cout << endl;

//PRINT THEM OUT

cout << "G + H = "; (G+H).Print(); cout << endl;

cout << "G - H = "; (G-H).Print(); cout << endl;

cout << "G \* H = "; (G\*H).Print(); cout << endl;

//OPERATIONS ON DIFFERENT TYPE

//WILL NOT WORK: CAN YOU MAKE IT WORK

//cout << "A + D = "; (A+D).Print(); cout << endl;

//cout << "A - G = "; (A-G).Print(); cout << endl;

//cout << "D \* G = "; (D\*G).Print(); cout << endl;

return 0;

}

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

**OUTPUT:**

A = { 3, 3 }{ 1 1 1 }{ 0 0 1 }{ 0 1 0 }

B = { 3, 3 }{ 0 3 3 }{ 0 2 1 }{ 2 3 2 }

C = { 3, 3 }{ 1 9 3 }{ 6 7 2 }{ 7 3 6 }

D = { 3, 3 }{ 3 4 2 }{ 2 7 0 }{ 5 6 3 }

E = { 3, 3 }{ 4 1 5 }{ 5 5 1 }{ 4 3 3 }

F = { 3, 3 }{ 1 3 0 }{ 0 1 2 }{ 3 1 2 }

G = { 3, 3 }{ 3.82698 1.2668 0.988759 }{ 2.34552 3.38353 3.51224 }{ 1.93697 1.13144 1.02916 }

H = { 3, 3 }{ 0.239393 0.284455 0.781635 }{ 0.975738 0.07015 0.364488 }{ 0.766147 0.09215 0.920103 }

I = { 3, 3 }{ 1.18053 0.0524068 0.476792 }{ 0.856053 1.44208 1.07452 }{ 1.7985 0.0666066 0.383149 }

A + B = { 3, 3 }{ 1 4 4 }{ 0 2 2 }{ 2 4 2 }

A - B = { 3, 3 }{ 1 -2 -2 }{ 0 -2 0 }{ -2 -2 -2 }

A \* B = { 3, 3 }{ 2 8 6 }{ 2 3 2 }{ 0 2 1 }

D + F = { 3, 3 }{ 4 7 2 }{ 2 8 2 }{ 8 7 5 }

D - F = { 3, 3 }{ -2 -1 -2 }{ -2 -6 2 }{ -2 -5 -1 }

D \* F = { 3, 3 }{ 9 25 2 }{ 12 19 6 }{ 21 31 12 }

G + H = { 3, 3 }{ 4.06637 1.55125 1.77039 }{ 3.32126 3.45368 3.87673 }{ 2.70312 1.22359 1.94926 }

G - H = { 3, 3 }{ 3.58759 0.982343 0.207124 }{ 1.36978 3.31338 3.14775 }{ 1.17083 1.03929 0.109056 }

G \* H = { 3, 3 }{ 2.90975 1.26858 4.36279 }{ 6.55384 1.2282 6.29822 }{ 2.35617 0.72519 2.87334 }

# Lab019: Implementation of a Matrix Chain Multiplication Algorithm(s).

The experiment requires you to create a class for handling of matrix chains. The program uses some of the classes/files already created. Namely:

1. myException.h
2. myRandom.h
3. myTimer.h
4. myArray.h
5. Matrix.h

The following are the sub-aims of this experiment.

1. Given a matrix chain, check if the matrices are compatible for multiplication or not.
2. If compatible, find the simplistic number of multiplication required.
3. Find the optimum sequene of multiplication that minimizes the number of multiplications. To do this implement the Recursive, the memoized top-down and the Bottom up algorithms.
4. Once the sequence is found using any of the techniques, do the multiplication of the cjhain and output the matrix.

The implementation framework is in the MatrixChain.h file as:

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

/\* FILE: myException.h

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

#ifndef MATRIXCHAIN\_H\_INCLUDED

#define MATRIXCHAIN\_H\_INCLUDED

#include <limits>

#include "Matrix.h"

#include <vector>

#include<fstream>

template <class T>

class MatrixChain {

std::vector<Matrix<T>> D; //The matrices

myArray<int> p; //The chain

Matrix<int> s; //The partition index holder matrix

Matrix<long> m; //The subchain cost matrix

public:

//CONSTRUCTORS

MatrixChain<T>() {}

MatrixChain<T>(Matrix<T> &x) {

D.push\_back(x);

}

MatrixChain<T>(MatrixChain<T> &x) {

if (x.empty()) {

for (auto i : x) D.push\_back(i);

}

}

MatrixChain<T>& push\_back(Matrix<T> &x){

D.push\_back(x);

return \*this;

}

//Auxiliary helpful functions

myArray<int> get\_p() { return p; }

Matrix<int> get\_s() { return s; }

Matrix<long> get\_m() { return m; }

//Chain Printing functions

friend std::istream& operator >> (std::istream& Input, MatrixChain<T>& x){

x.D.clear();

Matrix<T> a;

char c;

while (1) {

while (!Input.eof()) {

Input.get(c);

if (!isspace(c)) {

Input.putback(c);

break;

}

}

if (Input.eof()) break;

Input >> a;

x.D.push\_back(a);

}

return Input;

}

friend std::ostream& operator << (std::ostream& os, MatrixChain<T>& x){

for (size\_t i = 0; i< x.D.size(); i++) {

os << x.D[i];

}

return os;

}

void Print(std::ostream& os=std::cout){

for (size\_t i = 0; i< D.size(); i++) {

D[i].Print(os);

os << std::endl;

}

}

// If the matrices are compatible returns the array p or the size array.

// else returns an empty myArray object;

myArray<int> Compatible() {

for(size\_t i=0; i < D.size() - 1; i++)

if (D[i].Cols() != D[i+1].Rows()) myArray<int>();

myArray<int> a(D.size()+1);

a[0] = D[0].Rows();

for(int i=0; i < D.size(); i++) a[i+1] = D[i].Cols();

return a;

}

private:

// This function implements the Matrix Chain Order Calculation in a pure recursive fashion

// It returns the cost of multiplying I to j. This function is kept private

long MatrixOrder(int i, int j){

// YOUR IMPLEMENTATION HERE

}

public:

//Number of multiplications if it could find the sequence and -1 if the sequence is

// incompatible

long SequenceRecursive() {

p = Compatible();

if (p.Size() <= 0) return -1;

m.Clear();

s.Clear();

m = Matrix<long>(D.size(), D.size());

s = Matrix<int>(D.size(), D.size());

/\* COSMETIC ONLY \*/

for (int i = 0; i < s.Rows(); i++)

for (int j = 0; j < i; j++) {

m[i][j] = -1;

}

for (int i = 0; i < s.Rows(); i++)

for (int j = 0; j <= i; j++) {

s[i][j] = -1;

}

return MatrixOrder(1,D.size());

}

private:

// Number of multiplications if it could find the sequence and -1 if the sequence is

// incompatible

long LookupChain(int i, int j){

// YOUR IMPLEMENTATION HERE

}

public:

// Number of multiplications if it could find the sequence and -1 if the sequence is

// incompatible

// Memoized Top-Down, this is the public interface to the algorithm

long MemoizedSequence() {

p = Compatible();

if (p.Size() <= 0) return -1;

m.Clear();

s.Clear();

m = Matrix<long>(D.size(), D.size());

s = Matrix<int>(D.size(), D.size());

for (int i = 0; i < m.Rows(); i++)

for (int j = i; j < m.Cols(); j++) m[i][j] = LONG\_MAX;

return LookupChain(1,D.size());

}

// Number of multiplications if it could find the sequence and -1 if the sequence is

// incompatible

// Bottom-up

long Sequence() {

// YOUR IMPLEMENTATION HERE

}

private:

//private function used to print the optimum sequence

void PrintSequenceOrder(int i,int j,std::ostream& os=std::cout){

if (i == j) {

os << " A" << i << " ";

return;

}

os << "(";

PrintSequenceOrder(i,s[i][j],os);

PrintSequenceOrder(s[i][j]+1,j,os);

os << ")";

}

public:

//the public interface for printing the sequence

void PrintSequence(std::ostream& os=std::cout){

if (p.Size() <= 0) return;

PrintSequenceOrder(0,p.Size()-2);

}

private:

Matrix<T> Multiply(int i, int j){

if (i == j) return D[i];

return Multiply(i,s[i][j]) \* Multiply(s[i][j]+1,j);

}

public:

Matrix<T> Multiply(){

if (p.Size() <= 0) return Matrix<T>();

return Multiply(0,p.Size()-2);

}

};

#endif // MATRIXCHAIN\_H\_INCLUDED

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

A sample main.cpp file is given. Write your own main.cpp to demonstrate the working.

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

/\* FILE: main.cpp

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

#include <iostream>

#include <fstream>

#include "MatrixChain.h"

using namespace std;

#include "myTimer.h"

//NAME CLASH OCCURS FOR THE SYMBOL IN AND OUT DUE TO THIS CLASS

int main() {

myTimer T;

Matrix<long> A[10];

MatrixChain<long> B;

int p[] = {2, 17, 2, 25, 4, 12, 10, 17, 2, 25, 5 };

ofstream OUTFILE("Matrix.mat", ios::out | ios::trunc);

for (int i = 0; i<10; i++) {

A[i] = RandomMatrix<long>(p[i],p[i+1],2);

B.push\_back(A[i]);

}

OUTFILE << B;

OUTFILE.close();

MatrixChain<long> x;

ifstream INFILE("Matrix.mat");

INFILE >> x;

INFILE.close();

long COUNT;

T.StartTimer();

COUNT = x.SequenceRecursive();

T.EndTimer();

cout << "\nNumber of Multiplication (Recursive): " << COUNT

<< " Time Taken: " << T.GetInterval() << endl;

cout << "The sequence: ";

x.PrintSequence();

cout << "\nMultiply :";

x.Multiply().Print();

T.StartTimer();

COUNT = x.MemoizedSequence();

T.EndTimer();

cout << "\nNumber of Multiplication (Memoized): " << COUNT

<< " Time Taken: " << T.GetInterval() << endl;

cout << "The sequence: ";

x.PrintSequence();

cout << "\nMultiply :";

x.Multiply().Print();

T.StartTimer();

COUNT = x.Sequence();

T.EndTimer();

cout << "\nNumber of Multiplication (Bottom-up): " << COUNT

<< " Time Taken: " << T.GetInterval() << endl;

cout << "The sequence: ";

x.PrintSequence();

cout << "\nMultiply :";

x.Multiply().Print();

return 0;

}

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

**Output:**

Number of Multiplication (Recursive): 1238 Time Taken: 0.0003302

The sequence: ((( A0 A1 )(( A2 A3 )( A4 ( A5 ( A6 A7 )))))( A8 A9 ))

Multiply :{ 2, 5 }{ 398804 298052 258592 531456 283780 }{ 554203 414199 359360 738558 394361 }

Number of Multiplication (Memoized): 1238 Time Taken: 3.2e-05

The sequence: ((( A0 A1 )(( A2 A3 )( A4 ( A5 ( A6 A7 )))))( A8 A9 ))

Multiply :{ 2, 5 }{ 398804 298052 258592 531456 283780 }{ 554203 414199 359360 738558 394361 }

Number of Multiplication (Bottom-up): 1238 Time Taken: 3.21e-05

The sequence: ((( A0 A1 )(( A2 A3 )( A4 ( A5 ( A6 A7 )))))( A8 A9 ))

Multiply :{ 2, 5 }{ 398804 298052 258592 531456 283780 }{ 554203 414199 359360 738558 394361 }

# Lab020: Longest Common Subsequence

The aim is to represent sequences of items, generate all possible subsequence of the given subsequence and find the longest common subsequence of two sequences.

The following files are utilized:

1. myException.h
2. myRandom.h
3. myTimer.h
4. myArray.h
5. Matrix.h
6. Sequence.h (which is a new file)

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/ // FILE: Sequence.h

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

#ifndef SEQUENCE\_H\_INCLUDED

#define SEQUENCE\_H\_INCLUDED

#include <ostream>

#include <string>

#include <stack>

#include <vector>

#include "Matrix.h"

#include "myException.h"

#define MAX(a,b) ((a) > (b)) ? (a) : (b)

//Forward declarations

template <class T>

class Sequence;

template <class T>

class Sequence {

T \*D; //Data

int S; //Size

Matrix<long> C;

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

/\* A private function for allocation of memory \*/

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

void Allocate(int N) {

D = new T[S=N];

if (!D) throw myException("Sequence:Memory initialization failed\n");

}

public:

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

/\* CONSTRUCTOR(S) \*/

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

Sequence<T>(){ D = NULL; S = 0;} //noninitialized array

Sequence<T>(unsigned int Size) {

if (Size <= 0) {D = NULL; S = 0; return;}

Allocate(int(Size));

for (int i = 0; i < S; i++) D[i] =T(0);

}

Sequence<T>(T \*d, unsigned int Size) {

if (Size <= 0) {D = NULL; S = 0; return;}

Allocate(int(Size));

for (int i = 0; i < S; i++) D[i] =d[i];

}

Sequence<T>(const Sequence<T>& A) {

if (A.S <= 0) {D = NULL; S = 0; return;}

Allocate(A.S);

for (int i = 0; i < S; i++) D[i] =A.D[i];

}

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

/\* DESTRUCTOR \*/

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

virtual ~Sequence(){ Clear(); }

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

/\* Auxiliary functions but useful \*/

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

bool isEmpty() { return S == 0;} // If data exists false else true

int Size() {return S;} // Number of data items put in array

void Clear(){ delete [] D; S = 0; D = NULL;} // Truncates the array

Matrix<long> getC() { return C;}

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

/\* OPERATOR OVERLOADING \*/

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

T& operator[] (int i){ //Access by index (zero based)

if (i > -1 && i < S) return D[i];

throw myException("Sequence: Out of range access\n");

}

Sequence<T>& operator = (Sequence<T> x){ //asignment

if (x.Size() < 0) throw myException("Sequence: Assigning non initialized array\n");

Clear();

if (x.Size() == 0) return \*this;

Allocate(x.S);

for (int i = 0; i < S; i++) D[i] = x.D[i];

return \*this;

}

Sequence<T>& operator += (Sequence<T> x){//concatenation of a sequence

Sequence<T> temp(S+x.S);

for (int i = 0; i < S; i++) temp[i] = D[i];

for (int i = 0; i < x.S; i++) temp[i+S] = x[i];

\*this = temp;

return \*this;

}

Sequence<T>& operator += (T x){//concatenation of one element

Sequence<T> temp(S+1);

for (int i = 0; i < S; i++) temp[i] = D[i];

temp[S] = x;

\*this = temp;

return \*this;

}

Sequence<T> operator + (Sequence<T> x){//concatenation of a sequence

Sequence<T> temp(\*this);

temp += x;

return temp;

}

Sequence<T> operator + (T x){//concatenation of a sequence

Sequence<T> temp(\*this);

temp += x;

return temp;

}

bool operator == (Sequence<T> &x){//equality check

if (S != x.S ) return false;

for (int i = 0; i < S; i++) if (D[i] != x[i]) return false;

return true;

}

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

/\* RECURSIVE SUBSEQUENCE CHECK FUNCTION \*/

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

bool subSequenceRecursive(Sequence<T> &Y){

return checkSubSequenceRecursive(Y, Y.S - 1, S - 1);

}

// Return true if YES else false

bool checkSubSequenceRecursive(Sequence<T> &Y, long m, long n){

//YOUR CODE HERE

}

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

/\* RECURSIVE SUBSEQUENCE CHECK FUNCTION END \*/

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

/\* ITERATIVE SUBSEQUENCE CHECK FUNCTION \*/

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

// Return true if YES else false

bool subSequenceCheck(Sequence<T> &Y){

// YOUR CODE HERE

}

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

/\* ITERATIVE SUBSEQUENCE CHECK FUNCTION ENDS \*/

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

/\* GENERATE SUBSEQUENCE \*/

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

std::vector<Sequence<T>> generateSubSequence(){

std::vector<Sequence<T>> A;

A.push\_back(Sequence<T>());

for (int i = 0; i < S; i++){

size\_t currSize = A.size();

for (size\_t j = 0; j < currSize; j++) {

A.push\_back(A[j]+D[i]);

}

}

//If you want to remove the null subsequence uncomment the next line

//A.erase(A.begin(),A.begin()+1);

return A;

}

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

/\* GENERATE SUBSEQUENCE END \*/

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

/\* FIND LCS Recursive \*/

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

//m is size of x, and n is size of n

private:

long recursiveLCS(size\_t m, Sequence<T>& y, size\_t n){

// YOUR CODE HERE

}

public:

//returns the the longest subsequence

Sequence<T> recursiveLCS(Sequence<T> x){

C.Clear();

C = Matrix<long>(S+1,x.S+1);

for (size\_t i = 1; i <= S; i++)

for (size\_t j = 1; j <= x.S; j++) C[i][j] = recursiveLCS(i, x, j);

return getSubSequence(x);

}

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

/\* FIND LCS Recursive ENDS \*/

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

/\* FIND LCS Memoized \*/

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

//m is size of x, and n is size of n

private:

long memoizedLCS(size\_t m, Sequence<T>& y, size\_t n){

// YOUR CODE HERE

}

public:

//returns the longest subsequence

Sequence<T> memoizedLCS(Sequence<T> x){

C.Clear();

C = Matrix<long>(S+1,x.S+1);

for (size\_t i = 0; i <= S; i++)

for (size\_t j = 0; j <= x.S; j++) C[i][j] = -1;

for (size\_t i = 0; i <= S; i++)

for (size\_t j = 0; j <= x.S; j++) C[i][j] = memoizedLCS(i,x,j);

C[S][x.S] = memoizedLCS(S, x, x.S);

return getSubSequence(x);

}

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

/\* FIND LCS Memoized ENDS \*/

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

/\* FIND LCS BOTTOM UP \*/

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

Sequence<T> LCS(Sequence<T> x){

// YOUR CODE HERE

}

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

/\* RETURN THE SUBSEQUENCE \*/

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

Sequence<T> getSubSequence(Sequence<T> &y) {

// YOUR CODE HERE

}

//OUTPUT STREAM OVERLOAD. NOT A VERY TRICKY FUNCTION

friend std::ostream& operator << (std::ostream& os, Sequence<T> x){

if (x.Size() <= 0) return os << "{ Null }";

for (int i = 0; i < x.Size() - 1; i++ ) os << x[i] << " ";

return os << x[x.Size() - 1];

}

//OUTPUT in a slightly formatted manner

void Print(std::ostream& os=std::cout ){

if (Size() <= 0) {os << "{ Null }"; return;}

os << "{ ";

for (int i = 0; i < Size()-1; i++ ) os << D[i] << " ";

os << D[Size() - 1] << " }";

}

// Read an array. The size of the array is fixed before reading

friend std::istream& operator >> (std::istream& Input, Sequence<T>& x){

int n=x.S;

for (int i = 0; i< n; i++) Input >> x[i];

return Input;

}

};

#endif // SEQUENCE\_H\_INCLUDED

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

The sample main file is main.cpp

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

/\* FILE: main.cpp

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

#include <iostream>

#include <iomanip>

#include <vector>

#include "Sequence.h"

#include "myTimer.h"

using namespace std;

int main(){

char a[] = {'A','B','C','B','D','A','B'};

char b[] = {'B','D','C','A', 'B', 'A'};

char c[] = {'C','Q','D','A'};

Sequence<char> X(a,7);

Sequence<char> Y(b,6);

Sequence<char> Z(c,4);

cout << "X = " << X << endl;

cout << "Y = " << Y << endl;

cout << "Z = " << Z << endl;

cout << (X.subSequenceCheck(Y)? "YES": "NO") << endl;

cout << (X.subSequenceCheck(Z)? "YES": "NO") << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n";

vector<Sequence<char>> T = {X,Y,Z};

cout << T[1] << endl;

vector<Sequence<char>> U = Z.generateSubSequence();

for (int i = 0; i < U.size(); i++) {

cout << setw(-3) << i+1 << " = ";

U[i].Print();

cout << endl;

}

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n";

cout << "RECURSIVE:\n";

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n";

myTimer TIMER;

TIMER.StartTimer();

Sequence<char> S = X.recursiveLCS(Y);

TIMER.EndTimer();

cout << " X Y LCS Recursive: ";

S.Print();

cout << "\nTime Taken: " << TIMER.GetInterval() << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n";

cout << "Memoized:\n";

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n";

TIMER.StartTimer();

S = X.memoizedLCS(Y);

TIMER.EndTimer();

cout << " X Y LCS Memoized: ";

S.Print();

cout << "\nTime Taken: " << TIMER.GetInterval() << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n";

cout << "Bottom up:\n";

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n";

TIMER.StartTimer();

S = X.LCS(Y);

TIMER.EndTimer();

cout << " X Y LCS Bottom up: ";

S.Print();

cout << "\nTime Taken: " << TIMER.GetInterval() << endl;

return 0;

}

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

**OUTPUT:**

X = A B C B D A B

Y = B D C A B A

Z = C Q D A

NO

NO

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

B D C A B A

1 = { Null }

2 = { C }

3 = { Q }

4 = { C Q }

5 = { D }

6 = { C D }

7 = { Q D }

8 = { C Q D }

9 = { A }

10 = { C A }

11 = { Q A }

12 = { C Q A }

13 = { D A }

14 = { C D A }

15 = { Q D A }

16 = { C Q D A }

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

RECURSIVE:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

X Y LCS Recursive: { B D A B }

Time Taken: 0.0001916

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Memoized:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

X Y LCS Memoized: { B D A B }

Time Taken: 1.89e-05

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Bottom up:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

X Y LCS Bottom up: { B D A B }

Time Taken: 1.61e-05

# Lab021: Generation of all possible BST if keys are 1 to N

The aim is to generate all possible BST structure if the keys are in the range 1, 2, … , N.

The primary file is:

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

/\* FILE: Tree.h

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

#ifndef TREE\_H\_INCLUDED

#define TREE\_H\_INCLUDED

#include <cmath>

#include <iostream>

#include <ostream>

#include <vector>

#define MAX(a,b) ((a) > (b)) ? (a) : (b)

class Node{

int V;// Key Value

Node \*L; //LEFT CHILD POINTER

Node \*R; //RIGHT CHILD POINTER

public:

Node(): V(0), L(nullptr), R(nullptr) {}

Node(int x): V(x), L(nullptr), R(nullptr) {}

Node(int x, Node \*left, Node \*right): V(x), L(left), R(right) {}

Node(Node &x): V(x.V), L(x.L), R(x.R) {}

Node& operator = (Node x) {

V = x.V;

L = x.L;

R = x.R;

return \*this;

}

friend std::ostream& operator >> (std::ostream& os, Node &x) {

return os << x.V;

}

int& Value() { return V;}

Node\*\* Left() { return &L;}

Node\*\* Right() { return &R;}

};

class allBSTree{

std::vector<Node \*> D; //The BST's

allBSTree() {}

Node\* CreateNode(int v){

Node \*x = new Node;

x->Value() = v;

\*(x->Left()) = nullptr;

\*(x->Right()) = nullptr;

return x;

}

std::vector<Node \*> generateBST(long i, long j){

std::vector<Node \*> Temp;

if (i > j) Temp.push\_back(NULL);

for (long m = i; m <= j; m++) {

std::vector<Node\*> lst = generateBST(i, m - 1); // Left Sub tree

std::vector<Node\*> rst = generateBST(m + 1, j);

for (long n = 0; n < long(lst.size()); n++) { //j= n

Node\* lc = lst[n]; // left child

for (long k = 0; k < long(rst.size()); k++) {

Node\* rc = rst[k]; // right child

Node\* Data = CreateNode(m); // m as root

\*Data->Left() = lc; // left child subtree

\*Data->Right() = rc; // right child subtree

Temp.push\_back(Data); // add to D

}

}

}

return Temp;

}

public:

allBSTree(unsigned short n){

//DO NOT USE FOR MORE THAN 10, n is the number of keys

D = generateBST(1,n);

}

void makeBSTree(unsigned short n) {

D.clear();

D = generateBST(0,n-1);

}

unsigned long numberBST() {

return (unsigned long)(D.size());

}

private:

int treeHeight(Node\* Data){

if (Data == NULL) return 0;

int a = treeHeight(\*Data->Left());

int b = treeHeight(\*Data->Right());

return int(MAX(a,b)) + 1;

}

int Column(int A){

if (A == 1) return A;

return Column(A - 1) + Column(A - 1) + 1;

}

void pTree(int \*\*Layout, Node \*Data, int column, int row, int height) {

if (Data == NULL) return;

Layout[row][column] = Data->Value();

pTree(Layout, \*Data->Left(), column - pow(2, height - 2), row + 1, height - 1);

pTree(Layout, \*Data->Right(), column + pow(2, height - 2), row + 1, height - 1);

}

void Printer(Node \*Data, std::ostream& os = std::cout) {

int h = treeHeight(Data);

int column = Column(h);

int \*\*Layout = new int\*[h];

for (int i = 0; i < h; i++) Layout[i] = new int[column];

for (int i = 0; i < h; i++) {

for (int j = 0; j < column; j++) Layout[i][j] = 0;

}

pTree(Layout, Data, column / 2, 0, h);

for (int i = 0; i < h; i++) {

for (int j = 0; j < column; j++) {

if (Layout[i][j] == 0) std::cout << " " << " ";

else std::cout << Layout[i][j] << " ";

}

std::cout << std::endl;

}

}

void preorder(Node \*d){

if (d != NULL) {

std::cout << d->Value() << " ";

preorder(\*d->Left());

preorder(\*d->Right());

}

}

public:

void Print(unsigned long i, std::ostream& os = std::cout){//print ith tree

if (D.size() == 0 || i < 1 || i > D.size() ) {

os << "{ NULL }";

return;

}

Printer(D[i-1]);

}

void preorder(){

for (long i = 0; i < long(D.size()); i++) {

preorder(D[i]);

std::cout << std::endl;

}

}

long C(int n){//Catalan number

if (n < 0) return 0;

else if (n==0 || n == 1) return 1;

else return 2\*(2\*n-1) \* C(n-1) / (n+1);

}

};

#endif // TREE\_H\_INCLUDED

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

The sample main file is:

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

/\* FILE: main.cpp

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

#include <iostream>

#include "Tree.h"

using namespace std;

int main() {

int n = 5;

allBSTree x(n);

cout << x.numberBST() << endl;

for (int i = 1; i <= x.C(n); i++) {

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n";

cout << " i = " << i << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n";

x.Print(i);

}

return 0;

}

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

**OUTPUT:**

No of BST: 14

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

i = 1

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1

2

3

4

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

i = 2

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1

2

4

3

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

i = 3

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1

3

2 4

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

i = 4

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1

4

2

3

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

i = 5

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1

4

3

2

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

i = 6

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

2

1 3

4

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

i = 7

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

2

1 4

3

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

i = 8

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

3

1 4

2

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

i = 9

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

3

2 4

1

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

i = 10

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

4

1

2

3

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

i = 11

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

4

1

3

2

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

i = 12

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

4

2

1 3

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

i = 13

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

4

3

1

2

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

i = 14

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

4

3

2

1

# Lab022: Generation of all possible BST if Keys and associate probabilities are given.

The following is given:

1. All N Key values.
2. All N Key probabilities
3. All N+1 dummy probabilities are given (and if they are not given, they have to be assessed).

Given this information, design and implement code in C++ (as a generic class, as the Key values can be any comparable quantity).

The code must provide facility for:

1. Making the BST
2. Calculating the expected search cost
3. Printing the tree

The code uses the following files:

1. myException.h
2. myTimer.h

The principal file for implementation is the fiile below (the part to be implemented is marked):

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

/\* FILE: optimalBST.h

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

#ifndef OPTIMALBST\_H\_INCLUDED

#define OPTIMALBST\_H\_INCLUDED

#include <cmath>

#include <iostream>

#include<iomanip>

#include <ostream>

#include <vector>

#include <limits>

#include "myException.h"

#define MAX(a,b) ((a) > (b)) ? (a) : (b)

//T must be a comparable data type

template <class T>

class Node{

T V;// Key

int IDX; //The index of the key

double prob; //associated key probability (NOT A NECESSARY ITEM)

Node \*L; //LEFT CHILD POINTER

Node \*R; //RIGHT CHILD POINTER

public:

Node<T>(): L(nullptr), R(nullptr), IDX(-1) {}

Node<T>(T x): V(x), L(nullptr), R(nullptr), IDX(-1) {}

//Node<T>(T x, Node \*left, Node \*right): V(x), L(left), R(right) {}

Node<T>(Node<T> &x): V(x.V), L(x.L), R(x.R), IDX(x.IDX){}

virtual ~Node<T>() { }

Node<T>& operator = (Node<T> x) {

V = x.V;

L = x.L;

R = x.R;

IDX = x.IDX;

return \*this;

}

friend std::ostream& operator << (std::ostream& os, Node x) {

return os << x.V;

}

T& Value() { return V;}

int& Index() { return IDX;}

double& Probability() { return prob;}

Node<T>\*\* Left() { return &L;}

Node<T>\*\* Right() { return &R;}

};

typedef enum{RECURSIVE, MEMOIZED, ITERATIVE} bst\_mode;

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

/\* Assumptions: \*/

/\* 1. KEYS ARE ASSUMED TO BE SORTED IN ASCENDING ORDER \*/

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

template <class T>

class optimalBSTree{

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

/\* PRIVATE DATA \*/

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

Node<T> \* D; //The BST's

std::vector<std::vector<double>> W;

std::vector<std::vector<int>> R;

std::vector<std::vector<double>> C;

std::vector<double> P; //all key probabilities

std::vector<double> Q; //all dummy probabilities

std::vector<T> KEY; //all keys

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

/\* PRINTING FUNCTIONS \*/

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

private:

int treeHeight(Node<T>\* Data){

if (Data == NULL) return 0;

int a = treeHeight(\*Data->Left());

int b = treeHeight(\*Data->Right());

return int(MAX(a,b)) + 1;

}

int Column(int A){

if (A == 1) return A;

return Column(A - 1) + Column(A - 1) + 1;

}

void pTree(int \*\*Layout, Node<T>\* Data, int column, int row, int height) {

if (Data == NULL) return;

Layout[row][column] = Data->Index(); //Data->Value()

pTree(Layout, \*Data->Left(), column - pow(2, height - 2), row + 1, height - 1);

pTree(Layout, \*Data->Right(), column + pow(2, height - 2), row + 1, height - 1);

}

void Print(Node<T> \* Data, std::ostream& os = std::cout) {

int h = treeHeight(Data);

int column = Column(h);

int \*\*Layout = new int\*[h];

for (int i = 0; i < h; i++) Layout[i] = new int[column];

for (int i = 0; i < h; i++) {

for (int j = 0; j < column; j++) Layout[i][j] = -1;

}

pTree(Layout, Data, column / 2, 0, h);

for (int i = 0; i < h; i++) {

for (int j = 0; j < column; j++) {

if (Layout[i][j] == -1) std::cout << " " << " ";

else std::cout << KEY[Layout[i][j]] << " ";

}

std::cout << std::endl;

}

}

void preorder(Node<T> \*d){

if (d != NULL) {

std::cout << d->Value() << " ";

preorder(\*d->Left());

preorder(\*d->Right());

}

}

public:

void Print(std::ostream& os = std::cout){//print the tree

if (D == nullptr) {

os << "{ NULL }";

return;

}

Print(D);

}

void preorder(){

preorder(D);

std::cout << std::endl;

}

void PrintW(std::ostream &os = std::cout) {

for(size\_t i = 0; i <= W.size() - 1; i++) {

for(size\_t j = 0; j <=W[0].size() - 1; j++) {

os << std::setw(5) << W[i][j] << " ";

}

os << std::endl;

}

}

void PrintC(std::ostream &os = std::cout) {

for(size\_t i = 0; i <= C.size() - 1; i++) {

for(size\_t j = 0; j <=C[0].size() - 1; j++) {

os << std::setw(5) << C[i][j] << " ";

}

os << std::endl;

}

}

void PrintR(std::ostream &os = std::cout) {

for(size\_t i = 0; i <= R.size() - 1; i++) {

for(size\_t j = 0; j <=R[0].size() - 1; j++) {

os << std::setw(5) << R[i][j] << " ";

}

os << std::endl;

}

}

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

/\* PRINTING FUNCTIONS END \*/

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

/\* FUNCTIONS TO BE IMPLEMENTED: PRIVATE FUNCTIONS \*/

/\* These functions actually calculate the matrix C \*/

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

private:

double generateRecursive(long i, long j){

//static long NoOfCalls = 1;

//std::cout << NoOfCalls++ << std::endl;

//YOUR IMPLEMENTATION HERE

}

double generateMemoized(long i, long j){

//static long NoOfCalls = 1;

//std::cout << NoOfCalls++ << std::endl;

//YOUR IMPLEMENTATION HERE

}

double generateIterative(long n){

//static long NoOfCalls = 1;

//std::cout << NoOfCalls++ << std::endl;

//YOUR IMPLEMENTATION HERE

return C[0][n];

}

Node<T>\* makeBST(short i,short j){

Node<T> \*d;

if(i > j) return nullptr;

d = CreateNode(R[i][j]);

//Node<T> x = \*d;

\*d->Left() = makeBST(i, R[i][j] - 1); //left subtree

\*d->Right() = makeBST(R[i][j]+1, j); //right subtree

return d;

}

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

/\* PRIVATE FUNCTION TO DELETE THE TREE, THE PUBLIC INTERFACE IS OF THE SAME \*/

/\* NAME (OVERLOADED) \*/

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

void Clear(Node<T> \*t){

if (!t) return;

Clear(\*t->Left());

Clear(\*t->Right());

delete t;

}

public:

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

/\* CONSTRUCTORS AND DESTRUCTORS \*/

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

optimalBSTree() {D=NULL;}

optimalBSTree<T>(unsigned short n,T\* Key, double \*p, double \*q, bst\_mode x = ITERATIVE):D(nullptr){

makeBST(n,Key,p, q, x);

}

~optimalBSTree<T>() {Clear();}

//returns 0 if inconsistent, else the cost measure

double makeBST(unsigned short n,T \*Key, double \*p, double \*q, bst\_mode x = ITERATIVE){

Clear();

//first consistency check

if (!Key || !p) {

std::cout << "Error 001 in initialization\n";

return 0;

}

//create space for values

try{

C = std::vector<std::vector<double>> (n+1, std::vector<double>(n+1, 0));

W = std::vector<std::vector<double>> (n+1, std::vector<double>(n+1, 0));

R = std::vector<std::vector<int>> (n, std::vector<int>(n, -1));

KEY = std::vector<T>();

for (unsigned i = 0; i < n; i++) P.push\_back(p[i]);

if(q) for (unsigned i = 0; i <= n; i++) Q.push\_back(q[i]);

else Q = std::vector<double>(n+1,0);

for (unsigned i = 0; i < n; i++) KEY.push\_back(Key[i]);

}

catch(...){

std::cout << "Error 002 in initialization\n";

Clear();

return 0;

}

if(P.size() != n || P.size() != KEY.size() || P.size() != (Q.size()-1)) {

Clear();

std::cout << "Error 003 in initialization\n";

return 0;

}

for (int i = 0; i<n; i++)

if(P[i] < 0 || Q[i] < 0) {

Clear();

std::cout << "Error 004 in initialization\n";

return 0;

}

if (Q[n] < 0) {

Clear();

std::cout << "Error 005 in initialization\n";

return 0;

}

//consistency check over

//rationalization of probabilities

double sum = 0.0;

for (int i = 0; i<n; i++) sum += P[i]+Q[i];

sum += Q[n];

for (int i = 0; i<n; i++) {P[i] /= sum; Q[i] /= sum;}

Q[n] /=sum;

//rationalization of probabilities over

for (unsigned i = 0; i <= n; i++) {

C.at(i).at(i) = q[i];

W.at(i).at(i) = q[i];

}

if (x== RECURSIVE) generateRecursive(0,n);

else if (x== MEMOIZED) generateMemoized(0,n);

else if (x== ITERATIVE) generateIterative(n);

D = makeBST(0,n-1);

return C[0][n];

}

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

/\* USEFUL FUNCTIONS \*/

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

private:

Node<T>\* CreateNode(int idx){

Node<T> \*x = new Node<T>;

if (!x) throw myException("Memory allocation Error.");

x->Value() = KEY[idx];

x->Index() = idx;

x->Probability() = P[idx]; //REMOVE THIS IF THE FIELD prob NOT IN NODE

\*(x->Left()) = nullptr;

\*(x->Right()) = nullptr;

return x;

}

public:

std::vector<std::vector<double>> getW() { return W; }

std::vector<std::vector<int>> getR() { return R; }

std::vector<std::vector<double>> getC() { return C; }

void Clear() {

Clear(D);

KEY.clear();

P.clear();

Q.clear();

W.clear();

R.clear();

C.clear();

D=nullptr;

}

};

#endif // OPTIMALBST\_H\_INCLUDED

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

The sample main file is as follows:

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

/\* FILE: main.cpp

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

#include <iostream>

#include <iomanip>

#include "optimalBST.h"

#include "myTimer.h"

using namespace std;

int main()

{

//double p1[] = {0.15, 0.10, 0.05, 0.10, 0.20};

//double q1[] = {0.05, 0.10, 0.05, 0.05, 0.05, 0.10};

//string Key1[] = {"AA","AB","BA","BC","XYZ"};

double p2[] ={ 0.04, 0.06, 0.08, 0.02, 0.10, 0.12, 0.14};

double q2[] ={ 0.06, 0.06, 0.06, 0.06, 0.05, 0.05, 0.05, 0.05};

int Key2[] = {1,2,3,4,5,6,7};

myTimer TIMER;

optimalBSTree<int> A;

double Cost;

cout << "\n----------------------------------------------------------------------\n";

cout << "RECURSIVE: \n";

cout << "----------------------------------------------------------------------\n";

TIMER.StartTimer();

Cost = A.makeBST(7,Key2,p2,q2,RECURSIVE);

TIMER.EndTimer();

cout << " Time Taken: " << TIMER.GetInterval() << "(Seconds)\n";

cout << " Optimal Search Cost: " << Cost << "\n";

cout << "TREE:\n";

A.Print();

cout << "\n----------------------------------------------------------------------\n";

cout << "MEMOIZED: \n";

cout << "----------------------------------------------------------------------\n";

TIMER.StartTimer();

Cost = A.makeBST(7,Key2,p2,q2,MEMOIZED);

TIMER.EndTimer();

cout << " Time Taken: " << TIMER.GetInterval() << "(Seconds)\n";

cout << " Optimal Search Cost: " << Cost << "\n";

cout << "TREE:\n";

A.Print();

cout << "\n----------------------------------------------------------------------\n";

cout << "ITERATIVE: \n";

cout << "----------------------------------------------------------------------\n";

TIMER.StartTimer();

Cost = A.makeBST(7,Key2,p2,q2,ITERATIVE);

TIMER.EndTimer();

cout << " Time Taken: " << TIMER.GetInterval() << "(Seconds)\n";

cout << " Optimal Search Cost: " << Cost << "\n";

cout << "TREE:\n";

A.Print();

cout << "\n----------------------------------------------------------------------\n";

A.Clear();

A.Print();

return 0;

}

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

The output shall be as:

----------------------------------------------------------------------

RECURSIVE:

----------------------------------------------------------------------

Time Taken: 0.0001313(Seconds)

Optimal Search Cost: 3.12

TREE:

5

2 7

1 3 6

4

----------------------------------------------------------------------

MEMOIZED:

----------------------------------------------------------------------

Time Taken: 2.88e-05(Seconds)

Optimal Search Cost: 3.12

TREE:

5

2 7

1 3 6

4

----------------------------------------------------------------------

ITERATIVE:

----------------------------------------------------------------------

Time Taken: 3.04e-05(Seconds)

Optimal Search Cost: 3.12

TREE:

5

2 7

1 3 6

4

----------------------------------------------------------------------

{ NULL }

# Lab023: 0/1 Knap sack problem with exhaustive enumeration of feasible subsets of items

Generate all possible configurations (feasible to be shown) and find the optimal one by recursive enumeration of feasible sets for the 0/1 Knapsack Problem.

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

/\* FILE: myException.h

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

#ifndef MYEXCEPTION\_H\_INCLUDED

#define MYEXCEPTION\_H\_INCLUDED

#include <exception>

#include <string>

class myException: public std::exception {

protected:

/\* Error message \*/

std::string msg;

public:

/\* Constructor (C strings). \*/

explicit myException(const char\* message) : msg(message) {}

/\* Constructor (C++ STL strings). \*/

explicit myException(const std::string& message) : msg(message) {}

/\* Destructor. Virtual to allow for subclassing if required. \*/

virtual ~myException() noexcept {}

/\* Returns a pointer to the (constant) error description. \*/

virtual const char\* what() const noexcept { return msg.c\_str(); }

};

#endif // MYEXCEPTION\_H\_INCLUDED

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

/\* FILE: Item.h

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

#ifndef ITEM\_H\_INCLUDED

#define ITEM\_H\_INCLUDED

#include <ostream>

template <class T>

class Item{

T N; //The name of the item

double V; //Value

double W; //Weight

public:

Item<T>(): V(0),W(0) {}

Item<T>(T x, double value, double weight): N(x), V(value), W(weight){}

Item<T>(const Item<T> &x): N(x.N), V(x.V), W(x.W){}

virtual ~Item<T>() { }

Item<T>& operator = (const Item<T> &x) {

N = x.N;

V = x.V;

W = x.W;

return \*this;

}

friend bool operator < (Item<T> x, Item<T> y) { return x.N < y.N; }

friend bool operator > (Item<T> x, Item<T> y) { return x.N > y.N; }

friend bool operator == (Item<T> x, Item<T> y) { return x.N == y.N; }

friend bool operator >= (Item<T> x, Item<T> y) { return x.N >= y.N; }

friend bool operator <= (Item<T> x, Item<T> y) { return x.N <= y.N; }

friend bool operator != (Item<T> x, Item<T> y) { return x.N != y.N; }

friend std::ostream& operator << (std::ostream& os, Item<T> x) {

if (x.V == 0 || x.W == 0) return os << "NULL ( 0, 0 )";

return os << x.N << "( " << x.V << ", " << x.W << " )";

}

T& Name() { return N;}

double& Value() { return V;}

double& Weight() { return W;}

};

#endif // ITEM\_H\_INCLUDED

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

/\* FILE: KnapSack01Exhaustive.h

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

#ifndef KNAPSACK01EXHAUSTIVE\_H\_INCLUDED

#define KNAPSACK01EXHAUSTIVE\_H\_INCLUDED

#include <vector>

#include <set>

#include <ostream>

#include<limits>

#include <string>

#include "myException.h"

#include "Item.h"

//Prints a set of Items

template <typename T>

std::ostream& operator << (std::ostream& os, std::set<Item<T>> &x) {

os << "{\t\n" ;

for (auto it1=x.begin(); it1 != x.end(); ++it1) {

os << "\t" << \*it1 << std::endl;

}

os << "} ";

return os;

}

template <class T>

class KnapSack01Exhaustive{

std::vector<Item<T>> D; //The data

std::vector<std::set<Item<T>>> S; //valid subsets

std::vector<double> V; //value of each subset

std::vector<double> Wt; //weight of each subset

double W; //Knapsack maximum Weight

size\_t Idx; //Optimal set index;

public:

KnapSack01Exhaustive<T>(): W(-1.0){}

KnapSack01Exhaustive<T>(std::vector<Item<T>> x, double KnapSackWt):

D(x), W(KnapSackWt) {

generate();

findOptimalIndex();

}

KnapSack01Exhaustive<T>(const KnapSack01Exhaustive<T>& x): D(x.D), W(x.W) {

generate();

findOptimalIndex();

}

~KnapSack01Exhaustive() { clear(); }

void clear() {

S.clear();

D.clear();

V.clear();

Wt.clear();

Idx = -1;

W = -1.0;

}

std::vector<Item<T>>& Data(){ return D; }

std::vector<std::set<Item<T>>> allFeasible() {

generate();

findOptimalIndex();

return S;

}

size\_t optimalSetIndex() {//assumes that findOptimal has been called correctly

return Idx+1;

}

std::set<Item<T>>& optimalSet() {//assumes that findOptimal has been called correctly

return S[Idx];

}

double optimalValue() {//assumes that findOptimal has been called correctly

return V[Idx];

}

double optimalWt() {//assumes that findOptimal has been called correctly

return Wt[Idx];

}

double& KnapSackWeight() { return W;}

friend std::ostream& operator << (std::ostream& os, KnapSack01Exhaustive<T> &x) {

size\_t i = 0;

for (auto it=x.S.begin(); it != x.S.end(); ++it) {

os << "SET " << i + 1 << ": ";

os << \*it;

os << "Value = " << x.V[i] << " Wt = " << x.Wt[i] << std::endl << std::endl;

i++;

}

return os;

}

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

/\* Ideally the following functions should be private \*/

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

void generate(){

if (D.size() <= 0) throw myException("Data items not specified.\n");

if (W <= 0.0) throw myException("Knapsack weight not valid.\n");

for (size\_t i = 0; i < D.size(); i++) {

if (D[i].Weight() <= 0)

throw myException("Weight of atleast one item is invalid.\n");

if (D[i].Value() <= 0)

throw myException("Value of atleast item is invalid.\n");

}

// initial check completed

std::vector<Item<T>> d;

for (size\_t i = 0; i < D.size(); i++) if (D[i].Weight() <= W) d.push\_back(D[i]);

if (d.size() <= 0)

throw myException("No item with weight less than knapsack capacity\n");

// validity check completed

S.clear();

V.clear();

Wt.clear();

S.push\_back(std::set<Item<T>>()); //the null set

V.push\_back(0.0);

Wt.push\_back(0.0);

for (size\_t i = 0; i < d.size(); i++){

size\_t currSize = S.size();

for (size\_t j = 0; j < currSize; j++) {

double nWt = Wt[j] + d[i].Weight();

if (nWt <= W) {

std::set<Item<T>> a(S[j]);

a.insert(d[i]);

S.push\_back(a);

V.push\_back(V[j] + d[i].Value());

Wt.push\_back(nWt);

}

}

}

}

size\_t findOptimalIndex() {

double MAX = -std::numeric\_limits<double>::infinity();

for (size\_t it=0; it < S.size(); ++it)

if (V[it] > MAX) {

MAX = V[it];

Idx = it;

}

return Idx + 1;

}

};

#endif // KNAPSACK01EXHAUSTIVE\_H\_INCLUDED

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

The main file is:

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

/\* FILE: main.cpp

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

#include <iostream>

#include "Item.h"

#include <vector>

#include <string>

#include "KnapSack01Exhaustive.h"

using namespace std;

int main() {

//FIRST THE ITEMS

{

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n";

cout << "EXAMPLE 001\n";

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n";

Item<string> A(string("A"), 7000, 10);

Item<string> B(string("B"), 10000, 20);

Item<string> C(string("C"), 12000, 30);

Item<string> D(string("D"), 15000, 35);

double W = 50;

cout << A << endl;

Item<string> E(A);

cout << (E=B) << endl;

std::vector<Item<string>> d;

d.push\_back(A);

d.push\_back(B);

d.push\_back(C);

d.push\_back(D);

KnapSack01Exhaustive<string> KS01Exhaustive(d,W);

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n";

cout << KS01Exhaustive << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n";

cout << "Optimal Index: " << KS01Exhaustive.optimalSetIndex() << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n";

cout << "Optimal Set: " << KS01Exhaustive.optimalSet()

<< "Value = " << KS01Exhaustive.optimalValue()

<< " Wt = " << KS01Exhaustive.optimalWt() << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n";

}

{

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n";

cout << "EXAMPLE 002\n";

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n";

Item<string> A1(string("A"), 2, 3);

Item<string> A2(string("B"), 3, 5);

Item<string> A3(string("C"), 3, 7);

Item<string> A4(string("D"), 4, 4);

Item<string> A5(string("E"), 4, 3);

Item<string> A6(string("F"), 5, 9);

Item<string> A7(string("G"), 7, 2);

Item<string> A8(string("H"), 8, 11);

Item<string> A9(string("I"), 8, 5);

double W = 15;

std::vector<Item<string>> d;

d.push\_back(A1);

d.push\_back(A2);

d.push\_back(A3);

d.push\_back(A4);

d.push\_back(A5);

d.push\_back(A6);

d.push\_back(A7);

d.push\_back(A8);

d.push\_back(A9);

KnapSack01Exhaustive<string> KS01Exhaustive(d,W);

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n";

cout << KS01Exhaustive << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n";

cout << "Optimal Index: " << KS01Exhaustive.optimalSetIndex() << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n";

cout << "Optimal Set: " << KS01Exhaustive.optimalSet()

<< "Value = " << KS01Exhaustive.optimalValue()

<< " Wt = " << KS01Exhaustive.optimalWt() << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n";

}

return 0;

}

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

**The output is as:**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

EXAMPLE 001

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

A( 7000, 10 )

B( 10000, 20 )

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

SET 1: {

} Value = 0 Wt = 0

SET 2: {

A( 7000, 10 )

} Value = 7000 Wt = 10

SET 3: {

B( 10000, 20 )

} Value = 10000 Wt = 20

SET 4: {

A( 7000, 10 )

B( 10000, 20 )

} Value = 17000 Wt = 30

SET 5: {

C( 12000, 30 )

} Value = 12000 Wt = 30

SET 6: {

A( 7000, 10 )

C( 12000, 30 )

} Value = 19000 Wt = 40

SET 7: {

B( 10000, 20 )

C( 12000, 30 )

} Value = 22000 Wt = 50

SET 8: {

D( 15000, 35 )

} Value = 15000 Wt = 35

SET 9: {

A( 7000, 10 )

D( 15000, 35 )

} Value = 22000 Wt = 45

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Optimal Index: 7

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Optimal Set: {

B( 10000, 20 )

C( 12000, 30 )

} Value = 22000 Wt = 50

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

EXAMPLE 002

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

SET 1: {

} Value = 0 Wt = 0

SET 2: {

A( 2, 3 )

} Value = 2 Wt = 3

SET 3: {

B( 3, 5 )

} Value = 3 Wt = 5

SET 4: {

A( 2, 3 )

B( 3, 5 )

} Value = 5 Wt = 8

SET 5: {

C( 3, 7 )

} Value = 3 Wt = 7

SET 6: {

A( 2, 3 )

C( 3, 7 )

} Value = 5 Wt = 10

SET 7: {

B( 3, 5 )

C( 3, 7 )

} Value = 6 Wt = 12

SET 8: {

A( 2, 3 )

B( 3, 5 )

C( 3, 7 )

} Value = 8 Wt = 15

SET 9: {

D( 4, 4 )

} Value = 4 Wt = 4

SET 10: {

A( 2, 3 )

D( 4, 4 )

} Value = 6 Wt = 7

SET 11: {

B( 3, 5 )

D( 4, 4 )

} Value = 7 Wt = 9

SET 12: {

A( 2, 3 )

B( 3, 5 )

D( 4, 4 )

} Value = 9 Wt = 12

SET 13: {

C( 3, 7 )

D( 4, 4 )

} Value = 7 Wt = 11

SET 14: {

A( 2, 3 )

C( 3, 7 )

D( 4, 4 )

} Value = 9 Wt = 14

SET 15: {

E( 4, 3 )

} Value = 4 Wt = 3

SET 16: {

A( 2, 3 )

E( 4, 3 )

} Value = 6 Wt = 6

SET 17: {

B( 3, 5 )

E( 4, 3 )

} Value = 7 Wt = 8

SET 18: {

A( 2, 3 )

B( 3, 5 )

E( 4, 3 )

} Value = 9 Wt = 11

SET 19: {

C( 3, 7 )

E( 4, 3 )

} Value = 7 Wt = 10

SET 20: {

A( 2, 3 )

C( 3, 7 )

E( 4, 3 )

} Value = 9 Wt = 13

SET 21: {

B( 3, 5 )

C( 3, 7 )

E( 4, 3 )

} Value = 10 Wt = 15

SET 22: {

D( 4, 4 )

E( 4, 3 )

} Value = 8 Wt = 7

SET 23: {

A( 2, 3 )

D( 4, 4 )

E( 4, 3 )

} Value = 10 Wt = 10

SET 24: {

B( 3, 5 )

D( 4, 4 )

E( 4, 3 )

} Value = 11 Wt = 12

SET 25: {

A( 2, 3 )

B( 3, 5 )

D( 4, 4 )

E( 4, 3 )

} Value = 13 Wt = 15

SET 26: {

C( 3, 7 )

D( 4, 4 )

E( 4, 3 )

} Value = 11 Wt = 14

SET 27: {

F( 5, 9 )

} Value = 5 Wt = 9

SET 28: {

A( 2, 3 )

F( 5, 9 )

} Value = 7 Wt = 12

SET 29: {

B( 3, 5 )

F( 5, 9 )

} Value = 8 Wt = 14

SET 30: {

D( 4, 4 )

F( 5, 9 )

} Value = 9 Wt = 13

SET 31: {

E( 4, 3 )

F( 5, 9 )

} Value = 9 Wt = 12

SET 32: {

A( 2, 3 )

E( 4, 3 )

F( 5, 9 )

} Value = 11 Wt = 15

SET 33: {

G( 7, 2 )

} Value = 7 Wt = 2

SET 34: {

A( 2, 3 )

G( 7, 2 )

} Value = 9 Wt = 5

SET 35: {

B( 3, 5 )

G( 7, 2 )

} Value = 10 Wt = 7

SET 36: {

A( 2, 3 )

B( 3, 5 )

G( 7, 2 )

} Value = 12 Wt = 10

SET 37: {

C( 3, 7 )

G( 7, 2 )

} Value = 10 Wt = 9

SET 38: {

A( 2, 3 )

C( 3, 7 )

G( 7, 2 )

} Value = 12 Wt = 12

SET 39: {

B( 3, 5 )

C( 3, 7 )

G( 7, 2 )

} Value = 13 Wt = 14

SET 40: {

D( 4, 4 )

G( 7, 2 )

} Value = 11 Wt = 6

SET 41: {

A( 2, 3 )

D( 4, 4 )

G( 7, 2 )

} Value = 13 Wt = 9

SET 42: {

B( 3, 5 )

D( 4, 4 )

G( 7, 2 )

} Value = 14 Wt = 11

SET 43: {

A( 2, 3 )

B( 3, 5 )

D( 4, 4 )

G( 7, 2 )

} Value = 16 Wt = 14

SET 44: {

C( 3, 7 )

D( 4, 4 )

G( 7, 2 )

} Value = 14 Wt = 13

SET 45: {

E( 4, 3 )

G( 7, 2 )

} Value = 11 Wt = 5

SET 46: {

A( 2, 3 )

E( 4, 3 )

G( 7, 2 )

} Value = 13 Wt = 8

SET 47: {

B( 3, 5 )

E( 4, 3 )

G( 7, 2 )

} Value = 14 Wt = 10

SET 48: {

A( 2, 3 )

B( 3, 5 )

E( 4, 3 )

G( 7, 2 )

} Value = 16 Wt = 13

SET 49: {

C( 3, 7 )

E( 4, 3 )

G( 7, 2 )

} Value = 14 Wt = 12

SET 50: {

A( 2, 3 )

C( 3, 7 )

E( 4, 3 )

G( 7, 2 )

} Value = 16 Wt = 15

SET 51: {

D( 4, 4 )

E( 4, 3 )

G( 7, 2 )

} Value = 15 Wt = 9

SET 52: {

A( 2, 3 )

D( 4, 4 )

E( 4, 3 )

G( 7, 2 )

} Value = 17 Wt = 12

SET 53: {

B( 3, 5 )

D( 4, 4 )

E( 4, 3 )

G( 7, 2 )

} Value = 18 Wt = 14

SET 54: {

F( 5, 9 )

G( 7, 2 )

} Value = 12 Wt = 11

SET 55: {

A( 2, 3 )

F( 5, 9 )

G( 7, 2 )

} Value = 14 Wt = 14

SET 56: {

D( 4, 4 )

F( 5, 9 )

G( 7, 2 )

} Value = 16 Wt = 15

SET 57: {

E( 4, 3 )

F( 5, 9 )

G( 7, 2 )

} Value = 16 Wt = 14

SET 58: {

H( 8, 11 )

} Value = 8 Wt = 11

SET 59: {

A( 2, 3 )

H( 8, 11 )

} Value = 10 Wt = 14

SET 60: {

D( 4, 4 )

H( 8, 11 )

} Value = 12 Wt = 15

SET 61: {

E( 4, 3 )

H( 8, 11 )

} Value = 12 Wt = 14

SET 62: {

G( 7, 2 )

H( 8, 11 )

} Value = 15 Wt = 13

SET 63: {

I( 8, 5 )

} Value = 8 Wt = 5

SET 64: {

A( 2, 3 )

I( 8, 5 )

} Value = 10 Wt = 8

SET 65: {

B( 3, 5 )

I( 8, 5 )

} Value = 11 Wt = 10

SET 66: {

A( 2, 3 )

B( 3, 5 )

I( 8, 5 )

} Value = 13 Wt = 13

SET 67: {

C( 3, 7 )

I( 8, 5 )

} Value = 11 Wt = 12

SET 68: {

A( 2, 3 )

C( 3, 7 )

I( 8, 5 )

} Value = 13 Wt = 15

SET 69: {

D( 4, 4 )

I( 8, 5 )

} Value = 12 Wt = 9

SET 70: {

A( 2, 3 )

D( 4, 4 )

I( 8, 5 )

} Value = 14 Wt = 12

SET 71: {

B( 3, 5 )

D( 4, 4 )

I( 8, 5 )

} Value = 15 Wt = 14

SET 72: {

E( 4, 3 )

I( 8, 5 )

} Value = 12 Wt = 8

SET 73: {

A( 2, 3 )

E( 4, 3 )

I( 8, 5 )

} Value = 14 Wt = 11

SET 74: {

B( 3, 5 )

E( 4, 3 )

I( 8, 5 )

} Value = 15 Wt = 13

SET 75: {

C( 3, 7 )

E( 4, 3 )

I( 8, 5 )

} Value = 15 Wt = 15

SET 76: {

D( 4, 4 )

E( 4, 3 )

I( 8, 5 )

} Value = 16 Wt = 12

SET 77: {

A( 2, 3 )

D( 4, 4 )

E( 4, 3 )

I( 8, 5 )

} Value = 18 Wt = 15

SET 78: {

F( 5, 9 )

I( 8, 5 )

} Value = 13 Wt = 14

SET 79: {

G( 7, 2 )

I( 8, 5 )

} Value = 15 Wt = 7

SET 80: {

A( 2, 3 )

G( 7, 2 )

I( 8, 5 )

} Value = 17 Wt = 10

SET 81: {

B( 3, 5 )

G( 7, 2 )

I( 8, 5 )

} Value = 18 Wt = 12

SET 82: {

A( 2, 3 )

B( 3, 5 )

G( 7, 2 )

I( 8, 5 )

} Value = 20 Wt = 15

SET 83: {

C( 3, 7 )

G( 7, 2 )

I( 8, 5 )

} Value = 18 Wt = 14

SET 84: {

D( 4, 4 )

G( 7, 2 )

I( 8, 5 )

} Value = 19 Wt = 11

SET 85: {

A( 2, 3 )

D( 4, 4 )

G( 7, 2 )

I( 8, 5 )

} Value = 21 Wt = 14

SET 86: {

E( 4, 3 )

G( 7, 2 )

I( 8, 5 )

} Value = 19 Wt = 10

SET 87: {

A( 2, 3 )

E( 4, 3 )

G( 7, 2 )

I( 8, 5 )

} Value = 21 Wt = 13

SET 88: {

B( 3, 5 )

E( 4, 3 )

G( 7, 2 )

I( 8, 5 )

} Value = 22 Wt = 15

SET 89: {

D( 4, 4 )

E( 4, 3 )

G( 7, 2 )

I( 8, 5 )

} Value = 23 Wt = 14

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Optimal Index: 89

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Optimal Set: {

D( 4, 4 )

E( 4, 3 )

G( 7, 2 )

I( 8, 5 )

} Value = 23 Wt = 14

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# Lab024: Solve the 0/1 Knap sack problem using dynamic programming

The program shall utilize the following files:

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

/\* FILE: myException.h

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

#ifndef MYEXCEPTION\_H\_INCLUDED

#define MYEXCEPTION\_H\_INCLUDED

#include <exception>

#include <string>

class myException: public std::exception {

protected:

/\* Error message \*/

std::string msg;

public:

/\* Constructor (C strings). \*/

explicit myException(const char\* message) : msg(message) {}

/\* Constructor (C++ STL strings). \*/

explicit myException(const std::string& message) : msg(message) {}

/\* Destructor. Virtual to allow for subclassing if required. \*/

virtual ~myException() noexcept {}

/\* Returns a pointer to the (constant) error description. \*/

virtual const char\* what() const noexcept { return msg.c\_str(); }

};

#endif // MYEXCEPTION\_H\_INCLUDED

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

/\* FILE: myTimer.h

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

#ifndef MYTIMER\_H\_INCLUDED

#define MYTIMER\_H\_INCLUDED

#include <windows.h>

class myTimer{

LARGE\_INTEGER Frequency;

LARGE\_INTEGER startTime;

LARGE\_INTEGER endTime;

double interval;

public:

myTimer() { QueryPerformanceFrequency(&Frequency); }

void StartTimer(){ QueryPerformanceCounter(&startTime); }

void EndTimer(){ QueryPerformanceCounter(&endTime); }

double GetInterval() {

return (double) (endTime.QuadPart - startTime.QuadPart) / Frequency.QuadPart;

}

};

#endif // MYTIMER\_H\_INCLUDED

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

/\* FILE: Item.h

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

#ifndef ITEM\_H\_INCLUDED

#define ITEM\_H\_INCLUDED

#include <ostream>

template <class T, class U=int>

class Item{

T N; //The name of the item

double V; //Value

U W; //Weight

public:

Item<T,U>(): V(0),W(0) {}

Item<T,U>(T x, double value, U weight): N(x), V(value), W(weight){}

Item<T,U>(const Item<T,U> &x): N(x.N), V(x.V), W(x.W){}

virtual ~Item<T,U>() { }

Item<T,U>& operator = (const Item<T,U> &x) {

N = x.N;

V = x.V;

W = x.W;

return \*this;

}

friend bool operator < (Item<T,U> x, Item<T,U> y) { return x.N < y.N; }

friend bool operator > (Item<T,U> x, Item<T,U> y) { return x.N > y.N; }

friend bool operator == (Item<T,U> x, Item<T,U> y) { return x.N == y.N; }

friend bool operator >= (Item<T,U> x, Item<T,U> y) { return x.N >= y.N; }

friend bool operator <= (Item<T,U> x, Item<T,U> y) { return x.N <= y.N; }

friend bool operator != (Item<T,U> x, Item<T,U> y) { return x.N != y.N; }

friend std::ostream& operator << (std::ostream& os, Item<T,U> x) {

if (x.V == 0 || x.W == 0) return os << "NULL ( 0, 0 )";

return os << x.N << "( " << x.V << ", " << x.W << " )";

}

T& Name() { return N;}

double& Value() { return V;}

U& Weight() { return W;}

};

template <typename T, typename U=int>

bool compareItemByWt(Item<T,U> a, Item<T,U> b) {

return (a.Weight() < b.Weight());

}

template <typename T, typename U=int>

bool compareItemByValue(Item<T,U> a, Item<T,U> b) {

return (a.Value() < b.Value());

}

#endif // ITEM\_H\_INCLUDED

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

/\* FILE: KnapSack01.h

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

#ifndef KNAPSACK01\_H\_INCLUDED

#define KNAPSACK01\_H\_INCLUDED

#include <vector>

#include <set>

#include <ostream>

#include <iomanip>

#include<limits>

#include <string>

#include <algorithm>

#include "myException.h"

#include "Item.h"

//Prints a set of Items

template <typename T, typename U>

std::ostream& operator << (std::ostream& os, std::set<Item<T,U>> &x) {

os << "{\t\n" ;

for (auto it1=x.begin(); it1 != x.end(); ++it1) {

os << "\t" << \*it1 << std::endl;

}

os << "} ";

return os;

}

template <class T, class U=int>

class KnapSack01{

std::vector<Item<T,U>> D; //The data

std::set<Item<T,U>> S; //optimal set

std::vector<std::vector<double>> C; //cost metric

U W; //Knapsack maximum Weight

U Wt;//Weight of final

public:

KnapSack01<T,U>(): W(-1.0), Wt(0){}

KnapSack01<T,U>(std::vector<Item<T,U>> x, double KnapSackWt):

D(x), W(KnapSackWt),Wt(0) {

}

KnapSack01<T,U>(const KnapSack01<T,U>& x): D(x.D), W(x.W),Wt(0) {

}

~KnapSack01<T,U>() { clear(); }

void clear() {

S.clear();

C.clear();

D.clear();

Wt = 0.0;

W = -1.0;

}

std::vector<Item<T,U>>& Data(){ return D; }

//assumes that problem has been solved

double optimalValue() { return C[C.size()][C[0].size()]; }

//assumes that problem has been solved

double optimalWt() { return Wt; }

double& KnapSackWeight() { return W;}

//assumes that problem has been solved

friend std::ostream& operator << (std::ostream& os, KnapSack01<T> &x) {

if (x.S.size() > 0) os << x.S;

return os;

}

std::ostream& PrintC(std::ostream& os = std::cout){

if (C.size() == 0) return os << "{ NULL }" << std:: endl;

size\_t k = 0;

for (auto it = C.begin(); it != C.end(); ++it) {

if (k != 0 ) os << std::setw(4) << D[k-1].Name() << " ";

else os << std::setw(4) << " " << " ";

k++;

for (auto it1= it->begin(); it1 != it->end(); ++it1)

os << std::setw(3) << \*it1 << " ";

os << std::endl;

}

return os;

}

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

/\* RECURSIVE \*/

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

private:

double Recursive(int i, int j){ //returns C matrix

// YOUR IMPLEMENTATION HERE

}

public:

double Recursive() {

try{ Check(); }

catch(myException &x) {

std::cout << x.what() << std::endl;

std::cout << "Validity Check Failed." << std::endl;

return -1;

}

C = std::vector<std::vector<double>> (D.size()+1, std::vector<double>(W+1, 0));

for (size\_t i = 1; i <= D.size(); i++)

for (size\_t j = 1; j <= W; j++) Recursive(i,j);

Wt = Recursive(D.size(),W);

makeOptimal();

return Wt;

}

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

/\* MEMOIZED \*/

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

private:

double Memoized(int i, int j){

// YOUR IMPLEMENTATION HERE

return Wt = C[i][j];

}

public:

double Memoized() {

try{ Check(); }

catch(myException &x) {

std::cout << x.what() << std::endl;

std::cout << "Validity Check Failed." << std::endl;

return -1;

}

C = std::vector<std::vector<double>> (D.size()+1, std::vector<double>(W+1, 0));

for (size\_t i = 1; i <= D.size(); i++)

for (size\_t j = 1; j <= W; j++) Memoized(i,j);

Wt = Memoized(D.size(),W);

makeOptimal();

return Wt;

}

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

/\* BOTTOM UP: ITERATIVE \*/

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

public:

double Iterative() {

try{ Check(); }

catch(myException &x) {

std::cout << x.what() << std::endl;

std::cout << "Validity Check Failed." << std::endl;

return -1;

}

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

// COMPLETE THE CODE

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

makeOptimal();

return Wt;

}

private:

void Check() {

if (D.size() <= 0) throw myException("Data items not specified.\n");

if (W <= 0.0) throw myException("Knapsack weight not valid.\n");

for (size\_t i = 0; i < D.size(); i++) {

if (D[i].Weight() <= 0)

throw myException("Weight of atleast one item is invalid.\n");

if (D[i].Value() <= 0)

throw myException("Value of atleast one item is invalid.\n");

}

// initial check completed

//remove all infeasible items

auto itD = D.begin();

while (itD != D.end()) {

if(itD->Weight() > W) D.erase(itD);

++itD;

}

if (D.size() <= 0)

throw myException("No item with weight less than knapsack capacity\n");

// validity check completed

//sort by value

sort(D.begin(), D.end(), compareItemByValue<T>);

S.clear();

}

void makeOptimal(size\_t i, size\_t j){

if ( i <= 0 ) return;

if ( j <= 0 ) return;

double a = C[i][j];

while (a == C[i][j]) i--;

S.insert(D[i]);

j = j - D[i].Weight();

if ( std::abs(C[i][j]) < 1.0e-26) return; //check for zero double

makeOptimal(i,j);

}

void makeOptimal(){

size\_t i = D.size();

size\_t j = W;

S.clear();

makeOptimal(i,j);

}

};

#endif // KNAPSACK01\_H\_INCLUDED

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

The output from the program after you complete the implementation shall be as:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

EXAMPLE 001

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

A( 7000, 10 )

B( 10000, 20 )

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

KS01:Recursive

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Value = 19000

{

A( 7000, 10 )

C( 12000, 30 )

}

Time Taken = 5.89e-05(Seconds)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

KS01:Memoized

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Value = 19000

{

A( 7000, 10 )

C( 12000, 30 )

}

Time Taken = 4.06e-05(Seconds)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

KS01:Iterative

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Value = 19000

{

A( 7000, 10 )

C( 12000, 30 )

}

Time Taken = 1.71e-05(Seconds)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

EXAMPLE 002

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

KS01:Recursive

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Value = 23

{

D( 4, 4 )

E( 4, 3 )

G( 7, 2 )

I( 8, 5 )

}

Time Taken = 8.96e-05(Seconds)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

KS01:Memoized

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Value = 23

{

D( 4, 4 )

E( 4, 3 )

G( 7, 2 )

I( 8, 5 )

}

Time Taken = 4.85e-05(Seconds)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

KS01:Iterative

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Value = 23

{

D( 4, 4 )

E( 4, 3 )

G( 7, 2 )

I( 8, 5 )

}

Time Taken = 3.34e-05(Seconds)

# LAB025: Write a program to find the solution of the fractional Knapsack problem.

You are provided with a knapsack whose weight carrying capcity W is know. A list of items (ITEMS), Their associated weights (Wi) and the value of the item or profit from the item (Vi). Your code should treat items to be of any type while weights and profits in this case can be integral or floating point (but positive).

Demonstrate the working of your code on the data below. You may take W = 10Kg

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Items** | 1 | 2 | 3 | 4 | 5 |
| **Weights (in kg)** | 3 | 3 | 2 | 5 | 1 |
| **Profits / Value** | 10 | 15 | 10 | 12 | 8 |

1. Put your code for algorithm implementation here
2. Followed by the main.cpp file
3. Followed by the output.

# LAB026: Implement Huffman’s Encoding Algorithm using a tree

The following is given:

1. A set of alphabet Set S with N elements
2. The probability / frequencies of these set elements
3. Write a code that implements the Huffamn’s Encoding Algorithm and uses a tree structure

Do the following

1. Demonstrate the working of your code.
2. The code should be as generic as possible
3. Analyse the working of your code.
4. Generate and show the tree structure
5. Generate and print the codewords for each letter of the alphabet.

The data to be used are:

Data Set 001:

|  |  |  |
| --- | --- | --- |
| Alphabet | Frequency | Expected Code |
| space | 7 | 111 |
| a | 4 | 10 |
| e | 4 | 0 |
| f | 3 | 1101 |
| h | 2 | 1010 |
| i | 2 | 1000 |
| m | 2 | 111 |
| n | 2 | 10 |
| s | 2 | 1011 |
| t | 2 | 110 |
| l | 1 | 11001 |
| o | 1 | 110 |
| p | 1 | 10011 |
| r | 1 | 11000 |
| u | 1 | 111 |
| x | 1 | 10010 |

Data Set 002:

|  |  |  |  |
| --- | --- | --- | --- |
| Letter | Count | Letter | Frequency % |
| A | 14810 | A | 8.12 |
| B | 2715 | B | 1.49 |
| C | 4943 | C | 2.71 |
| D | 7874 | D | 4.32 |
| E | 21912 | E | 12.02 |
| F | 4200 | F | 2.30 |
| G | 3693 | G | 2.03 |
| H | 10795 | H | 5.92 |
| I | 13318 | I | 7.31 |
| J | 188 | J | 0.10 |
| K | 1257 | K | 0.69 |
| L | 7253 | L | 3.98 |
| M | 4761 | M | 2.61 |
| N | 12666 | N | 6.95 |
| O | 14003 | O | 7.68 |
| P | 3316 | P | 1.82 |
| Q | 205 | Q | 0.11 |
| R | 10977 | R | 6.02 |
| S | 11450 | S | 6.28 |
| T | 16587 | T | 9.10 |
| U | 5246 | U | 2.88 |
| V | 2019 | V | 1.11 |
| W | 3819 | W | 2.09 |
| X | 315 | X | 0.17 |
| Y | 3853 | Y | 2.11 |
| Z | 128 | Z | 0.07 |

# LAB027: Implement Huffman’s Encoding Algorithm using a Min-Heap

The fillowing is given:

1. A set of alphabet Set S with N elements
2. The probability / frequencies of these set elements
3. Write a code that implements the Huffamn’s Encoding Algorithm and uses a Min-Heap tree structure

Do the following

1. Demonstrate the working of your code.
2. The code should be as generic as possible
3. Analyse the working of your code.
4. Generate and show the tree structure
5. Generate and print the codewords for each letter of the alphabet.

Use the data set for LAB026.

# LAB028: Implement the graph Abstract Data Type using an adjacency matrix (undirected graphs).

The graph Abstract Data Type (ADT) is defined in terms of the operations that should be implemented. The operations required are defined later, first consider the following:

1. A graph (undirected graph, or a simple graph) is a pair G = (V, E), where V is a set whose elements are called vertices (nodes), and E is a set of paired vertices, whose elements are called edges (arcs, links or lines).

2. The vertices x and y of an edge (x, y) are called the endpoints of the edge. The edge is said to join x and y and to be incident on x and y. A vertex may belong to no edge, in which case it is not joined to any other vertex.

3. A null graph or an empty graph is a graph with no vertex (and no edge).

4. A vertex has a label (and other associated values).

5. An edge has two end points which are vertices of the graph. The two vertices allowed are distinct (self loop is not allowed).

6. Every link (edge) between two vertices has an associated weight (double value).

7. Sometimes the associated weight for no link or no edge between two vertices is represented by the value 0 (for absence of a connection) or ∞ (infinity) to represent that the cost of traversal of the edge is infinite. When the value 0 is used to represent the absence of the corresponding edge, the adjacency matrix is called the (normal) adjacency matrix, while if ∞ (infinity) to represent the cost of traversal of the edge, then the adjacency matrix is called the modified adjacency matrix. Your program should support the creation of both type of graph adjacency matrix representation.

The following operations must be supported:

**1. Specifying which type of graph adjacency matrix is to be used (NORMAL/MODIFIED). For adjacency list representation, you may not need explicit incorporation of this feature.**

**2. Creation of a null graph**

**3. Creation of a graph with one node / vertex.**

**4. Addition of nodes to the graph. (Label specified)**

**5. Deletion of a node from the graph. (Label specified)**

**6. Addition of edges to a graph. The endpoint vertices should already exist in the vertex list. The default weight is 1**

**7. Deletion of an edge from a graph. (Labels of end point specified or end points specified)**

**8. Find a vertex in a graph.**

**9. Check if the graph contains an edge (specified by the two end points).**

10. Find the number of nodes in the graph.

11. Find the number of edges in a graph.

12. Modify the weight of an edge (specified by the two end points). The default weight is 1

13. Modify the label of a node (old and new labels specified)

14. Find the indegree of a node (label specified)

15. Find the outdegree of a node (label specified)

16. Destroy a graph.

17. Save the graph to a file (First print the total number of vertices on the first line, followed by the vertex labels on separate line followed by a well formatted adjacency matrix layout/edge list).

18. Read the graph from a file (See 17)

19. Print the adjacency matrix/list (matrix if representation is an adjacency matrix and edge list if an adjacency list representation is used).

20. Print the Vertex list.

21. Print the edge list (each edge to be represented as (x,y,w), where x and y are the labels of the nodes connected by the edge and w is the weight. Only connections (not 0 or infinity) are to be printed.).

**22. Transpose the graph**

23. Convert a normal adjacency matrix to modified and vice versa.

24. Check for graph equality.

25 Assign a graph to another (A=B)

Implement the ADT as a class for undirected graphs. Demonstrate your working for a graph with at least 10 nodes and 15 edges. Show all operations.

Analyse the complexity of each function of your implementation.

# LAB029: Implement the graph Abstract Data Type using an adjacency matrix (directed graphs).

Implement the ADT given in LAB028 for directed graphs using adjacency matrix. Demonstrate your working for a graph with at least 10 nodes and 15 edges. Show all operations.

Analyse the complexity of each function of your implementation.

# LAB030: Implement the graph Abstract Data Type using an adjacency list (undirected graphs).

Implement the ADT given in LAB028 for undirected graphs using adjacency list. In this case edge with zero (effective) weights need not be stored.

Demonstrate your working for a graph with at least 10 nodes and 15 edges. Show all operations.

Analyse the complexity of each function of your implementation.

# LAB031: Implement the graph Abstract Data Type using an adjacency list (directed graphs).

Implement the ADT given in LAB028 for undirected graphs using adjacency list. In this case edge with zero (effective) weights need not be stored.

Demonstrate your working for a graph with at least 10 nodes and 15 edges. Show all operations.

Analyse the complexity of each function of your implementation.

# LAB032: Implement the Breadth First Search algorithm on a graph (adjacency matrix representation).

Implement the graph breadth first search algorithm. The implementation should take an instanance of any of the Graph implemented in LAB028 / LAB029 (Adjacency matrix representation). The following should be provided:

1. Given a source label, print the (shortest) path to all nodes reachable from it.

2. Print the BFS forest. In this case no source label is provided.

3. Find path between two nodes specified by their labels and return the path length;

# LAB033: Implement the Breadth First Search algorithm on a graph (adjacency list representation).

Implement the graph breadth first search algorithm. The implementation should take an instanance of any of the Graph implemented in LAB030 / LAB031 (Adjacency list representation). The following should be provided:

1. Given a source label, print the (shortest) path to all nodes reachable from it.

2. Print the BFS forest. In this case no source label is provided.

3. Find path between two nodes specified by their labels and return the path length;

# LAB034: Implement the Depth First Search algorithm on a graph (adjacency matrix representation).

Implement the graph depth first search algorithm. The implementation should take an instanance of any of the Graph implemented in LAB028 / LAB029 (Adjacency matrix representation). The following should be provided:

1. Given a source label, print the path to all nodes reachable from it.

2. Print the DFS forest. In this case no source label is provided.

3. Find path between two nodes specified by their labels and return the path length;

# LAB035: Implement the Depth First Search algorithm on a graph (adjacency list representation).

Implement the graph depth first search algorithm. The implementation should take an instanance of any of the Graph implemented in LAB030 / LAB031 (Adjacency list representation). The following should be provided:

1. Given a source label, print the path to all nodes reachable from it.

2. Print the DFS forest. In this case no source label is provided.

3. Find path between two nodes specified by their labels and return the path length;

# LAB036: Implement the Topological Sort algorithm on a graph (adjacency matrix representation).

Implement the topological sort algorithm. The implementation should take an instanance of any of the Graph implemented in LAB028 / LAB029 (Adjacency matrix representation) and the DFS implementation of LAB034. The following should be provided:

1. Do the topological sort or identify that the graph is not acyclic.

# LAB037: Implement the Topological Sort algorithm on a graph (adjacency list representation).

Implement the topological sort algorithm. The implementation should take an instanance of any of the Graph implemented in LAB030 / LAB031 (Adjacency list representation) and the DFS implementation of LAB035. The following should be provided:

1. Do the topological sort or identify that the graph is not acyclic.

# LAB038: Implement the Strongly Connected Component algorithm on a graph (adjacency matrix representation).

Implement the graph SCC algorithm. The implementation should take an instanance of any of the Graph implemented in LAB028 / LAB029 (Adjacency matrix representation) and the DFS implementation of LAB034. The following should be provided:

1. Find the SCC and print the trees.

# LAB039: Implement the Strongly Connected Component algorithm on a graph (adjacency list representation).

Implement the SCC algorithm. The implementation should take an instanance of any of the Graph implemented in LAB030 / LAB031 (Adjacency list representation) and the DFS implementation of LAB035. The following should be provided:

1. Find the SCC and print the trees.

# LAB040: Implement the Kruskal’s and Prim’s minimum spanning tree algorithm on a graph (adjacency matrix representation).

Implement the Kruskal’s and Prim’s minimum spanning tree algorithm. The implementation should take an instanance of any of the Graph implemented in LAB028 / LAB029 (Adjacency matrix representation). The following should be provided:

1. Find the minimum spanning tree(s) and print them.

# LAB041: Implement the Kruskal’s and Prim’s minimum spanning tree algorithm on a graph (adjacency list representation).

Implement the Kruskal’s and Prim’s minimum spanning tree algorithm. The implementation should take an instanance of any of the Graph implemented in LAB030 / LAB031 (Adjacency list representation). The following should be provided:

1. Find the minimum spanning tree(s) and print them.

# LAB042: Implement the Bellman-Ford’s and Dijikstra’s single source shortest path algorithm on a graph (adjacency matrix representation).

Implement the Bellman-Ford’s and Dijikstra’s single source shortestpath algorithm. The implementation should take an instanance of any of the Graph implemented in LAB028 / LAB029 (Adjacency matrixre presentation). The following should be provided:

1. Find the the paths and print them.

# LAB043: Implement the Bellman-Ford’s and Dijikstra’s single source shortest path algorithm on a graph (adjacency list representation).

Implement the Bellman-Ford’s and Dijikstra’s single source shortestpath algorithm. The implementation should take an instanance of any of the Graph implemented in LAB030 / LAB031 (Adjacency list representation). The following should be provided:

1. Find the the paths and print them.

# LAB044: Implement the Flloyd-Warshall’s all pair shortest path algorithm on a graph (adjacency matrix representation).

Implement the Fllyod – Warshall’s all pairs shortes tpath algorithm. The implementation should take an instanance of any of the Graph implemented in LAB028 / LAB029 (Adjacency matrixre presentation). The following should be provided:

1. Find the the paths and print them and the distances.

# LAB045: Implement the Flloyd-Warshall’s all pair shortest path algorithm on a graph (adjacency list representation).

Implement the Fllyod – Warshall’s all pairs shortes tpath algorithm. The implementation should take an instance of any of the Graph implemented in LAB030 / LAB031 (Adjacency list representation). The following should be provided:

1. Find the the paths and print them and the distances.

# LAB046: Implement the naïve (sub) string matching algorithm.

Objective:

Given a string of size N called S, and another string of size M called P, with N >= M, find the location of the first instance of P in S.

# LAB047: Implement the Rabin Karp (sub)string matching algorithm.

Objective:

Given a string of size N called S, and another string of size M called P, with N >= M, find the location of the first instance of P in S.