// Nwakwuo Moses

// CSDP 250: Project 2

// Program to convert a number to Binary

#include <iostream>

#include "decimalStack.h"

#include "decimalQueue.h"

void convertToBinary(double, int);//Function Definition for Conversion function

using namespace std;

int main(){

//Beginning of Program

cout << "Welcome to Decimal to Binary Program!\n";

cout << "This program collects any positve number and converts it to decimal.\n";

cout << "(Note: For Fractions, you will have to indicate the number of decimal places)\n";

double UserNumber = -1; //Initialize with negative number so while condition runs successfully.

//Input for the Number

while (UserNumber <= 0) {

cout << "Please input a positive number: ";

cin >> UserNumber;

if (UserNumber < 0) {

cout << "Invalid Input: Only positive numbers are accepted.\n";

}

}

int decimal\_place = 0;

//Input for the decimal Place

while (decimal\_place < 1 || decimal\_place > 10) {

cout << "Please input a positive number for decimal places (1~10): ";

cin >> decimal\_place;

if (decimal\_place < 1 || decimal\_place > 10) {

cout << "Invalid Input: Decimal places should be between 1 and 10.\n";

}

}

convertToBinary(UserNumber, decimal\_place);//Call conversion and output function

cout << "Thank you for using the program!\n";

return 0;

}

void convertToBinary(double number, int decimal\_places) {

BinaryStack wholeStack\_List;

BinaryQueue decimalQueue;

// Separate whole and decimal parts

int integerPart = number;

double decimalPart = number - integerPart;

// Convert whole part to binary using the stack

if (integerPart == 0) {

wholeStack\_List.push(0);

}

else {

wholeStack\_List.pileBinaryDigits(wholeStack\_List, integerPart);

}

// Convert decimal part to binary using the recursive queue function

if (decimalPart > 0) {

decimalQueue.afterDecimalQueue(decimalQueue, decimalPart, decimal\_places);

}

// Output the binary representation

cout << "Binary representation of " << number << " is = ";

// Output whole part

while (!wholeStack\_List.isEmpty()) {

cout << wholeStack\_List.pop();

}

// Output fractional part

if (decimalPart > 0) {

cout << ".";

while (!decimalQueue.isEmpty()) {

cout << decimalQueue.dequeue();

}

}

cout << endl;

}