**Title: PROGRAM FOR AUTOMATION.**

#include <iostream>

#include <string>

#include <algorithm>

#include <vector>

#include <cmath>

#include <unordered\_map>

// NumberConverter class handles all number system conversions.

class NumberConverter {

public:

// Function to convert from any base (2, 8, 10, 16) to decimal.

// It takes the number as a string and its base as an integer.

long long toDecimal(const std::string& input, int base) {

long long decimal = 0;

long long power = 1;

// Create a copy to reverse without modifying the original input string.

std::string s = input;

std::reverse(s.begin(), s.end());

// Iterate through each character (digit) of the reversed string.

for (char c : s) {

int digit;

if (base == 16) {

// For hexadecimal, look up the digit's value in the map.

if (hexToDecimalMap.count(c)) {

digit = hexToDecimalMap.at(c);

} else {

// Handle invalid hexadecimal digits.

std::cerr << "Error: Invalid hexadecimal digit '" << c << "'\n";

return -1;

}

} else {

// For other bases, convert character to integer.

digit = c - '0';

}

// Validate that the digit is valid for the given base.

if (digit < 0 || digit >= base) {

std::cerr << "Error: Invalid digit '" << c << "' for base " << base << "\n";

return -1;

}

// Add the digit's value multiplied by the corresponding power of the base.

decimal += digit \* power;

// Increase the power for the next digit.

power \*= base;

}

return decimal;

}

// Function to convert a decimal number to any base (2, 8, 16).

// It takes a decimal number and the target base as integers.

std::string fromDecimal(long long decimal, int base) {

if (decimal == 0) {

return "0";

}

std::string result = "";

// Use the remainder and division method to convert.

while (decimal > 0) {

int remainder = decimal % base;

if (base == 16) {

// For hexadecimal, use the map to get the character for the remainder.

result += decimalToHexMap.at(remainder);

} else {

// For other bases, convert the remainder to a string.

result += std::to\_string(remainder);

}

decimal /= base;

}

// The result is built in reverse order, so we reverse it to get the correct number.

std::reverse(result.begin(), result.end());

return result;

}

private:

// Maps for hexadecimal conversions, making the code cleaner and more readable.

std::unordered\_map<char, int> hexToDecimalMap = {

{'0', 0}, {'1', 1}, {'2', 2}, {'3', 3}, {'4', 4}, {'5', 5}, {'6', 6}, {'7', 7}, {'8', 8}, {'9', 9},

{'A', 10}, {'B', 11}, {'C', 12}, {'D', 13}, {'E', 14}, {'F', 15},

{'a', 10}, {'b', 11}, {'c', 12}, {'d', 13}, {'e', 14}, {'f', 15}

};

std::unordered\_map<int, char> decimalToHexMap = {

{0, '0'}, {1, '1'}, {2, '2'}, {3, '3'}, {4, '4'}, {5, '5'}, {6, '6'}, {7, '7'}, {8, '8'}, {9, '9'},

{10, 'A'}, {11, 'B'}, {12, 'C'}, {13, 'D'}, {14, 'E'}, {15, 'F'}

};

};

// Helper function to handle the common input/output and conversion flow.

void performConversion(NumberConverter& converter, int fromBase, int toBase, const std::string& fromName, const std::string& toName) {

std::string input;

std::cout << "Enter " << fromName << " number: ";

std::cin >> input;

// First convert to decimal.

long long decimalValue = converter.toDecimal(input, fromBase);

if (decimalValue != -1) {

// If conversion to decimal was successful, convert from decimal to the target base.

std::string result = converter.fromDecimal(decimalValue, toBase);

std::cout << toName << " equivalent: " << result << "\n";

}

}

int main() {

NumberConverter converter;

int choice;

std::string input;

do {

// Display the user menu.

std::cout << "\nNumber System Converter\n";

std::cout << "-----------------------\n";

std::cout << "1. Binary to Decimal\n";

std::cout << "2. Decimal to Binary\n";

std::cout << "3. Octal to Decimal\n";

std::cout << "4. Decimal to Octal\n";

std::cout << "5. Hexadecimal to Decimal\n";

std::cout << "6. Decimal to Hexadecimal\n";

std::cout << "7. Binary to Octal\n";

std::cout << "8. Octal to Binary\n";

std::cout << "9. Binary to Hexadecimal\n";

std::cout << "10. Hexadecimal to Binary\n";

std::cout << "11. Octal to Hexadecimal\n";

std::cout << "12. Hexadecimal to Octal\n";

std::cout << "0. Exit\n";

std::cout << "Enter your choice: ";

std::cin >> choice;

long long decimalValue;

std::string result;

// Handle user's choice using a switch statement.

switch (choice) {

case 1:

performConversion(converter, 2, 10, "binary", "Decimal");

break;

case 2:

performConversion(converter, 10, 2, "decimal", "Binary");

break;

case 3:

performConversion(converter, 8, 10, "octal", "Decimal");

break;

case 4:

performConversion(converter, 10, 8, "decimal", "Octal");

break;

case 5:

performConversion(converter, 16, 10, "hexadecimal", "Decimal");

break;

case 6:

performConversion(converter, 10, 16, "decimal", "Hexadecimal");

break;

case 7:

performConversion(converter, 2, 8, "binary", "Octal");

break;

case 8:

performConversion(converter, 8, 2, "octal", "Binary");

break;

case 9:

performConversion(converter, 2, 16, "binary", "Hexadecimal");

break;

case 10:

performConversion(converter, 16, 2, "hexadecimal", "Binary");

break;

case 11:

performConversion(converter, 8, 16, "octal", "Hexadecimal");

break;

case 12:

performConversion(converter, 16, 8, "hexadecimal", "Octal");

break;

case 0:

std::cout << "Exiting program.\n";

break;

default:

std::cout << "Invalid choice. Please try again.\n";

break;

}

} while (choice != 0); // Loop until the user chooses to exit.

return 0;

}