#include <iostream>

#include <string>

#include <cctype>

#include <string\_view>

static constexpr char HEX\_CHARS[] = "0123456789ABCDEF";

// Recorta y colapsa múltiples espacios en uno

std::string normalize(std::string\_view sv) {

size\_t start = 0, end = sv.size();

while (start < end && std::isspace(sv[start])) ++start;

while (end > start && std::isspace(sv[end - 1])) --end;

std::string out;

out.reserve(end - start);

bool prev\_space = false;

for (size\_t i = start; i < end; ++i) {

bool curr\_space = std::isspace(sv[i]);

if (curr\_space) {

if (prev\_space) continue;

out.push\_back(' ');

} else {

out.push\_back(sv[i]);

}

prev\_space = curr\_space;

}

return out;

}

// Texto plano -> Hexadecimal separado por espacios

std::string textoAHex(std::string\_view txt) {

size\_t n = txt.size();

if (n == 0) return "";

std::string out;

out.reserve(n \* 3 - 1);

for (size\_t i = 0; i < n; ++i) {

unsigned char c = static\_cast<unsigned char>(txt[i]);

out.push\_back(HEX\_CHARS[c >> 4]);

out.push\_back(HEX\_CHARS[c & 0xF]);

if (i + 1 < n) out.push\_back(' ');

}

return out;

}

// Hexadecimal (pares de dígitos, espacios ignorados) -> Texto plano

std::string hexATexto(std::string\_view hx) {

std::string out;

out.reserve(hx.size() / 2);

int val = 0, nibble = 0;

for (char ch : hx) {

if (std::isspace(ch)) continue;

ch = std::toupper(static\_cast<unsigned char>(ch));

int d = (ch >= '0' && ch <= '9') ? ch - '0'

: (ch >= 'A' && ch <= 'F') ? ch - 'A' + 10

: -1;

if (d < 0) {

std::cerr << "Byte inválido: '" << ch << "', se omite.\n";

continue;

}

val = (nibble == 0) ? (d << 4) : (val | d);

if (nibble == 1) {

out.push\_back(static\_cast<char>(val));

nibble = val = 0;

} else {

nibble = 1;

}

}

if (nibble == 1) {

// Si quedó medio byte, lo almacenamos

out.push\_back(static\_cast<char>(val));

}

return out;

}

int main() {

std::ios::sync\_with\_stdio(false);

std::cin.tie(nullptr);

std::string line;

std::cout << "=== Terminal Hex <-> Texto (optimizado) ===\n"

"Usos:\n"

" hex <texto>\n"

" text <hex>\n"

" exit\n"

"------------------------------------------\n";

while (true) {

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

if (!std::getline(std::cin, line)) break;

// Normalizar espacios y eliminar extremos

std::string cmdline = normalize(line);

if (cmdline.empty()) continue;

// Separar comando y argumento

size\_t sp = cmdline.find(' ');

std::string cmd = (sp == std::string::npos) ? cmdline

: cmdline.substr(0, sp);

std::string arg = (sp == std::string::npos) ? ""

: cmdline.substr(sp + 1);

if (cmd == "exit") {

std::cout << "Saliendo...\n";

break;

}

else if (cmd == "hex") {

std::cout << textoAHex(arg) << '\n';

}

else if (cmd == "text") {

std::cout << hexATexto(arg) << '\n';

}

else {

std::cout << "Comando desconocido: " << cmd << '\n';

}

}

return 0;

}