#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <curl/curl.h>

#include <libxml/HTMLparser.h>

struct MemoryStruct {

char \*memory;

size\_t size;

};

static size\_t WriteMemoryCallback(void \*contents, size\_t size, size\_t nmemb, void \*userp) {

size\_t realsize = size \* nmemb;

struct MemoryStruct \*mem = (struct MemoryStruct \*)userp;

char \*ptr = realloc(mem->memory, mem->size + realsize + 1);

if (ptr == NULL) {

printf("Not enough memory (realloc returned NULL)\n");

return 0;

}

mem->memory = ptr;

memcpy(&(mem->memory[mem->size]), contents, realsize);

mem->size += realsize;

mem->memory[mem->size] = 0;

return realsize;

}

char\* fetch\_html(const char \*url) {

CURL \*curl\_handle;

CURLcode res;

struct MemoryStruct chunk;

chunk.memory = malloc(1);

chunk.size = 0;

curl\_global\_init(CURL\_GLOBAL\_ALL);

curl\_handle = curl\_easy\_init();

curl\_easy\_setopt(curl\_handle, CURLOPT\_URL, url);

curl\_easy\_setopt(curl\_handle, CURLOPT\_WRITEFUNCTION, WriteMemoryCallback);

curl\_easy\_setopt(curl\_handle, CURLOPT\_WRITEDATA, (void \*)&chunk);

curl\_easy\_setopt(curl\_handle, CURLOPT\_USERAGENT, "libcurl-agent/1.0");

res = curl\_easy\_perform(curl\_handle);

if (res != CURLE\_OK) {

fprintf(stderr, "curl\_easy\_perform() failed: %s\n", curl\_easy\_strerror(res));

return NULL;

}

curl\_easy\_cleanup(curl\_handle);

curl\_global\_cleanup();

return chunk.memory;

}

void parse\_html(const char \*html) {

htmlDocPtr doc = htmlReadMemory(html, strlen(html), NULL, NULL, HTML\_PARSE\_NOERROR | HTML\_PARSE\_NOWARNING);

if (doc == NULL) {

fprintf(stderr, "Error: could not parse HTML\n");

return;

}

xmlNode \*root\_element = xmlDocGetRootElement(doc);

void parse\_node(xmlNode \*node) {

for (xmlNode \*cur\_node = node; cur\_node; cur\_node = cur\_node->next) {

if (cur\_node->type == XML\_ELEMENT\_NODE && strcmp((char \*)cur\_node->name, "div") == 0) {

xmlChar \*class = xmlGetProp(cur\_node, (const xmlChar \*)"class");

if (class && strcmp((char \*)class, "product") == 0) {

xmlNode \*child = cur\_node->children;

char product\_name[256] = {0};

char product\_price[256] = {0};

char product\_reviews[256] = {0};

for (xmlNode \*c = child; c; c = c->next) {

if (c->type == XML\_ELEMENT\_NODE) {

xmlChar \*class\_name = xmlGetProp(c, (const xmlChar \*)"class");

if (class\_name && strcmp((char \*)class\_name, "product-name") == 0) {

strcpy(product\_name, (char \*)xmlNodeGetContent(c));

} else if (class\_name && strcmp((char \*)class\_name, "product-price") == 0) {

strcpy(product\_price, (char \*)xmlNodeGetContent(c));

} else if (class\_name && strcmp((char \*)class\_name, "product-reviews") == 0) {

strcpy(product\_reviews, (char \*)xmlNodeGetContent(c));

}

xmlFree(class\_name);

}

}

printf("Product Name: %s\n", product\_name);

printf("Product Price: %s\n", product\_price);

printf("Product Reviews: %s\n\n", product\_reviews);

}

xmlFree(class);

}

parse\_node(cur\_node->children);

}

}

parse\_node(root\_element);

xmlFreeDoc(doc);

xmlCleanupParser();

}

int main(void) {

const char \*url = "file://path/to/your/sample.html";

char \*html = fetch\_html(url);

if (html) {

parse\_html(html);

free(html);

}

return 0;

}