// Resource class: represents various types of academic materials

class Resource {

public:

// Pure virtual functions to be implemented by derived classes

virtual std::string getTitle() const = 0;

virtual std::string getAuthor() const = 0;

virtual std::string getISBN() const = 0;

virtual std::string getType() const = 0;

};

// Book class: inherits from Resource

class Book : public Resource {

private:

std::string title;

std::string author;

std::string isbn;

public:

// Constructor to initialize book properties

Book(const std::string& title, const std::string& author, const std::string& isbn)

: title(title), author(author), isbn(isbn) {}

// Implement getTitle() function

std::string getTitle() const override { return title; }

// Implement getAuthor() function

std::string getAuthor() const override { return author; }

// Implement getISBN() function

std::string getISBN() const override { return isbn; }

// Implement getType() function

std::string getType() const override { return "Book"; }

};

// Journal class: inherits from Resource

class Journal : public Resource {

private:

std::string title;

std::string author;

std::string isbn;

public:

// Constructor to initialize journal properties

Journal(const std::string& title, const std::string& author, const std::string& isbn)

: title(title), author(author), isbn(isbn) {}

// Implement getTitle() function

std::string getTitle() const override { return title; }

// Implement getAuthor() function

std::string getAuthor() const override { return author; }

// Implement getISBN() function

std::string getISBN() const override { return isbn; }

// Implement getType() function

std::string getType() const override { return "Journal"; }

};

// DigitalResource class: inherits from Resource

class DigitalResource : public Resource {

private:

std::string title;

std::string author;

std::string url;

public:

// Constructor to initialize digital resource properties

DigitalResource(const std::string& title, const std::string& author, const std::string& url)

: title(title), author(author), url(url) {}

// Implement getTitle() function

std::string getTitle() const override { return title; }

// Implement getAuthor() function

std::string getAuthor() const override { return author; }

// Implement getISBN() function (returns empty string for digital resources)

std::string getISBN() const override { return ""; }

// Implement getType() function

std::string getType() const override { return "Digital Resource"; }

};

// Catalog class: manages the resource collection

class Catalog {

private:

// STL container to store unique pointers to resources

std::vector<std::unique\_ptr<Resource>> resources;

public:

// Add a new resource to the catalog

void addResource(std::unique\_ptr<Resource> resource) { resources.push\_back(std::move(resource)); }

// Remove a resource from the catalog by ISBN

void removeResource(const std::string& isbn) {

resources.erase(std::remove\_if(resources.begin(), resources.end(),

[&isbn](const auto& resource) { return resource->getISBN() == isbn; }),

resources.end());

}

// Search for resources by multiple criteria

std::vector<Resource\*> searchResources(const std::string& title = "", const std::string& author = "",

const std::string& isbn = "", const std::string& type = "") {

std::vector<Resource\*> result;

for (auto& resource : resources) {

if ((title.empty() || resource->getTitle() == title) &&

(author.empty() || resource->getAuthor() == author) &&

(isbn.empty() || resource->getISBN() == isbn) &&

(type.empty() || resource->getType() == type)) {

result.push\_back(resource.get());

}

}

return result;

}

// Generate a report of all resources, grouped by type

void generateReport() {

std::map<std::string, std::vector<Resource\*>> report;

for (auto& resource : resources) {

report[resource->getType()].push\_back(resource.get());

}

for (auto& pair : report) {

std::cout << pair.first << ":\n";

for (auto& resource : pair.second) {

std::cout << "Title: " << resource->getTitle() << ", Author: " << resource->getAuthor()

<< ", ISBN: " << resource->getISBN() << "\n";

}

}

}

};

// Command-line interface

int main() {