SOURCE CODE:

```
#include <iostream>
#include <string>
#include <exception>
#include <thread>
#include <future>
#include <vector>
#include <mutex>
#include <condition_variable>
#include <memory>

using namespace std;

class QuestionBase {
public:
virtual void askQuestion() = 0;
virtual vector<string> getOptions() = 0;
virtual ~QuestionBase() {}
};

template <typename T>
string getFunctionName() {
return typeid(T).name();
}

class Question1 : public QuestionBase {
public:
void askQuestion() override {
cout << "How are you feeling? Options: 1) bad, 2) happy, 3) sad" << endl;
}
vector<string> getOptions() override {
return {"bad", "happy", "sad"};
}
};

class Question2 : public QuestionBase {
public:
void askQuestion() override {
cout << "What is the capital of Japan, India, and the US? Options: 1) Tokyo, 2) New Delhi,
3) Washington D.C." << endl;
}
vector<string> getOptions() override {
return {"Tokyo", "New Delhi", "Washington D.C."};
}
};

class Question3 : public QuestionBase {
public:
void askQuestion() override {
```

```
cout << "What is the most common technology? Options: 1) AI, 2) ML, 3) Information Technology" << endl;
}
vector<string> getOptions() override {
return {"AI", "ML", "Information Technology"};
}
};

class Question4 : public QuestionBase {
public:
void askQuestion() override {
cout << "What is the tallest mountain in the world? Options: 1) K2, 2) Everest, 3) Kangchenjunga" << endl;
}
vector<string> getOptions() override {
return {"K2", "Everest", "Kangchenjunga"};
}
};

class Question5 : public QuestionBase {
public:
void askQuestion() override {
cout << "What is the fastest land animal? Options: 1) Cheetah, 2) Lion, 3) Gazelle" << endl;
}
vector<string> getOptions() override {
return {"Cheetah", "Lion", "Gazelle"};
}
};

class Question6 : public QuestionBase {
public:
void askQuestion() override {
cout << "What is the largest ocean? Options: 1) Atlantic, 2) Indian, 3) Pacific" << endl;
}
vector<string> getOptions() override {
return {"Atlantic", "Indian", "Pacific"};
}
};

class Question7 : public QuestionBase {
public:
void askQuestion() override {
cout << "Who wrote 'Hamlet'? Options: 1) Chaucer, 2) Shakespeare, 3) Dickens" << endl;
}
vector<string> getOptions() override {
return {"Chaucer", "Shakespeare", "Dickens"};
}
};

class Question8 : public QuestionBase {
```

```
public:
void askQuestion() override {
cout << "What is the speed of light? Options: 1) 300,000 km/s, 2) 150,000 km/s, 3) 450,000
km/s" << endl;
}
vector<string> getOptions() override {
return {"300,000 km/s", "150,000 km/s", "450,000 km/s"};
}
};

class Question9 : public QuestionBase {
public:
void askQuestion() override {
cout << "What is the smallest planet in our solar system? Options: 1) Earth, 2) Mars, 3)
Mercury" << endl;
}
vector<string> getOptions() override {
return {"Earth", "Mars", "Mercury"};
}
};

class Question10 : public QuestionBase {
public:
void askQuestion() override {
cout << "What is the largest mammal? Options: 1) Elephant, 2) Blue Whale, 3) Giraffe" <<
endl;
}
vector<string> getOptions() override {
return {"Elephant", "Blue Whale", "Giraffe"};
}
};

class Question11 : public QuestionBase {
public:
void askQuestion() override {
cout << "What is the hardest natural substance? Options: 1) Gold, 2) Diamond, 3) Silver" <<
endl;
}
vector<string> getOptions() override {
return {"Gold", "Diamond", "Silver"};
}
};

class Question12 : public QuestionBase {
public:
void askQuestion() override {
cout << "What is the smallest country in the world? Options: 1) Monaco, 2) Vatican City, 3)
San Marino" << endl;
}
vector<string> getOptions() override {
```

```
return {"Monaco", "Vatican City", "San Marino"};
}
};

class Question13 : public QuestionBase {
public:
void askQuestion() override {
cout << "What is the largest desert in the world? Options: 1) Sahara, 2) Gobi, 3) Antarctic"
<< endl;
}
vector<string> getOptions() override {
return {"Sahara", "Gobi", "Antarctic"};
}
};

class Question14 : public QuestionBase {
public:
void askQuestion() override {
cout << "What is the longest river in the world? Options: 1) Nile, 2) Amazon, 3) Yangtze" <<
endl;
}
vector<string> getOptions() override {
return {"Nile", "Amazon", "Yangtze"};
}
};

class Question15 : public QuestionBase {
public:
void askQuestion() override {
cout << "What is the main ingredient in guacamole? Options: 1) Tomato, 2) Avocado, 3)
Pepper" << endl;
}
vector<string> getOptions() override {
return {"Tomato", "Avocado", "Pepper"};
}
};

class AnswerBase {
public:
virtual void giveAnswer(const string& answer) = 0;
virtual ~AnswerBase() {}
};

class Answer1 : public AnswerBase {
public:
void giveAnswer(const string& answer) override {
cout << "You chose: " << answer << endl;
}
};
```

```
class Answer2 : public AnswerBase {
public:
void giveAnswer(const string& answer) override {
cout << "The capital of Japan is Tokyo, India is New Delhi, and the US is Washington D.C."
<< endl;
}
};

class Answer3 : public AnswerBase {
public:
void giveAnswer(const string& answer) override {
cout << "The most common technology is " << answer << endl;
}
};

class Answer4 : public AnswerBase {
public:
void giveAnswer(const string& answer) override {
cout << "The tallest mountain in the world is " << answer << endl;
}
};

class Answer5 : public AnswerBase {
public:
void giveAnswer(const string& answer) override {
cout << "The fastest land animal is " << answer << endl;
}
};

class Answer6 : public AnswerBase {
public:
void giveAnswer(const string& answer) override {
cout << "The largest ocean is the " << answer << endl;
}
};

class Answer7 : public AnswerBase {
public:
void giveAnswer(const string& answer) override {
cout << "Hamlet was written by " << answer << endl;
}
};

class Answer8 : public AnswerBase {
public:
void giveAnswer(const string& answer) override {
cout << "The speed of light is " << answer << endl;
}
};
```

```
class Answer9 : public AnswerBase {
public:
void giveAnswer(const string& answer) override {
cout << "The smallest planet in our solar system is " << answer << endl;
}
};

class Answer10 : public AnswerBase {
public:
void giveAnswer(const string& answer) override {
cout << "The largest mammal is the " << answer << endl;
}
};

class Answer11 : public AnswerBase {
public:
void giveAnswer(const string& answer) override {
cout << "The hardest natural substance is " << answer << endl;
}
};

class Answer12 : public AnswerBase {
public:
void giveAnswer(const string& answer) override {
cout << "The smallest country in the world is " << answer << endl;
}
};

class Answer13 : public AnswerBase {
public:
void giveAnswer(const string& answer) override {
cout << "The largest desert in the world is the " << answer << endl;
}
};

class Answer14 : public AnswerBase {
public:
void giveAnswer(const string& answer) override {
cout << "The longest river in the world is the " << answer << endl;
}
};

class Answer15 : public AnswerBase {
public:
void giveAnswer(const string& answer) override {
cout << "The main ingredient in guacamole is " << answer << endl;
}
};

class QuestionException : public exception {
```

```
string message;
public:
QuestionException(const string& msg) : message(msg) {}
const char* what() const noexcept override {
return message.c_str();
}
};

void askAndAnswer(promise<string> && p, QuestionBase* question, condition_variable&
cv, mutex& mtx, bool& ready) {
try {
question->askQuestion();
vector<string> options = question->getOptions();
int choice;
cin >> choice;

if (choice < 1 || choice > options.size()) {
throw QuestionException("Invalid choice");
}

string answer = options[choice - 1];

{
lock_guard<mutex> lock(mtx);
p.set_value(answer);
ready = true;
}

cv.notify_one();
} catch (const exception & e) {
p.set_exception(make_exception_ptr(QuestionException("Invalid question asked")));
cv.notify_one();
}
}

void processAnswer(future<string> && f, AnswerBase* answer, condition_variable& cv,
mutex& mtx, bool& ready) {
try {
unique_lock<mutex> lock(mtx);
cv.wait(lock, [&ready] { return ready; });
string result = f.get();
cout << "Processing answer: " << result << endl;
answer->giveAnswer(result);
} catch (const exception & e) {
cerr << e.what() << endl;
}
}

int main() {
vector<unique_ptr<QuestionBase>> questions;
```

```
questions.emplace_back(make_unique<Question1>());
questions.emplace_back(make_unique<Question2>());
questions.emplace_back(make_unique<Question3>());
questions.emplace_back(make_unique<Question4>());
questions.emplace_back(make_unique<Question5>());
questions.emplace_back(make_unique<Question6>());
questions.emplace_back(make_unique<Question7>());
questions.emplace_back(make_unique<Question8>());
questions.emplace_back(make_unique<Question9>());
questions.emplace_back(make_unique<Question10>());
questions.emplace_back(make_unique<Question11>());
questions.emplace_back(make_unique<Question12>());
questions.emplace_back(make_unique<Question13>());
questions.emplace_back(make_unique<Question14>());
questions.emplace_back(make_unique<Question15>());

vector<unique_ptr<AnswerBase>> answers;
answers.emplace_back(make_unique<Answer1>());
answers.emplace_back(make_unique<Answer2>());
answers.emplace_back(make_unique<Answer3>());
answers.emplace_back(make_unique<Answer4>());
answers.emplace_back(make_unique<Answer5>());
answers.emplace_back(make_unique<Answer6>());
answers.emplace_back(make_unique<Answer7>());
answers.emplace_back(make_unique<Answer8>());
answers.emplace_back(make_unique<Answer9>());
answers.emplace_back(make_unique<Answer10>());
answers.emplace_back(make_unique<Answer11>());
answers.emplace_back(make_unique<Answer12>());
answers.emplace_back(make_unique<Answer13>());
answers.emplace_back(make_unique<Answer14>());
answers.emplace_back(make_unique<Answer15>());

mutex mtx;
condition_variable cv;
bool ready = false;

int userChoice;

cout << "Choose a question to answer (1 to 15):" << endl;
cout << "1. How are you feeling?" << endl;
cout << "2. What is the capital of Japan, India, and the US?" << endl;
cout << "3. What is the most common technology?" << endl;
cout << "4. What is the tallest mountain in the world?" << endl;
cout << "5. What is the fastest land animal?" << endl;
cout << "6. What is the largest ocean?" << endl;
cout << "7. Who wrote 'Hamlet'?" << endl;
cout << "8. What is the speed of light?" << endl;
cout << "9. What is the smallest planet in our solar system?" << endl;
cout << "10. What is the largest mammal?" << endl;
```

```
cout << "11. What is the hardest natural substance?" << endl;
cout << "12. What is the smallest country in the world?" << endl;
cout << "13. What is the largest desert in the world?" << endl;
cout << "14. What is the longest river in the world?" << endl;
cout << "15. What is the main ingredient in guacamole?" << endl;
cin >> userChoice;

if (userChoice >= 1 && userChoice <= 15) {
promise<string> p;
future<string> f = p.get_future();

thread qThread(askAndAnswer, move(p), questions[userChoice - 1].get(), ref(cv), ref(mtx),
ref(ready));
thread aThread(processAnswer, move(f), answers[userChoice - 1].get(), ref(cv), ref(mtx),
ref(ready));

qThread.join();
aThread.join();

ready = false;
} else {
cout << "Invalid choice" << endl;
}

return 0;
}
```