1021

Please answer the following c++ questions. Do not simply answer yes/no.

Always explain your assumptions and reasoning.

1. memset is sometimes used to initialize data in a constructor like the example below. What is the benefit of initializing this way? Does it work in this example? Does it work in general ? Is it a good idea in general?

class A {

public:

A();

private:

int a;

float f;

char str[35];

long \*lp;

};

A::A()

{

memset(this, 0, sizeof(\*this));

}

Benefit: All variables are initialized to zero in one statement and much faster than individual variable initializations.

It doesn't work well for char array/string as there are 2 problems:

1) Null terminator character is '\0' not 0,

2) Only first element of char array needs to be initialized to '\0', here we waste cpu cycles initializing all 35 with 0.

Hence in general, it is better to initialize each variable separately. Plus there will be no risk of incorrect default initialization when new member variable is added.

1. Your task is to implement the Reduce function using templates. The Reduce fn applies a function of two arguments cumulatively to the items of an STL container, from begin() to end(), so as to reduce the sequence to a single value. For example, Reduce(<list containing 1,2,3,4,5>, std::plus<int>()) should calculate ((((1+2)+3)+4)+5).

class NotEnoughElements {};

template <typename Container, typename Function>

typename Container::value\_type

Reduce(const Container& c, Function fn) throw (NotEnoughElements)

{

typename Container::const\_iterator it = c.begin();

if(it == c.end()) {

throw new NotEnoughElements;

}

typename Container::value\_type sum = \*it;

while(++it != c.end()) {

sum = fn(sum, \*it);

}

return sum;

}

1. Write a C++ program that would find and print the first longest ascending or descending contiguous subsequence for a vector of integers. For example, given a vector with

4, 2, 1, 2, 3, 4, 3, 5, 1, 2, 4, 6, 5

the program would find the underlined subsequence and print it.

void print\_longest\_contiguous(const vector<int> &v) {

size\_t start(0), end(0), max\_start(0), max\_end(0);

for(size\_t i(1); i < v.size(); ++i) {

int difference = v[i] - v[i - 1];

if(difference \* difference == 1) {

int direction = (v[end] - v[start]) \* difference;

end = i;

if(direction <= 0) {

start = i - 1;

}

if(max\_end - max\_start < end - start) {

max\_start = start;

max\_end = end;

}

} else {

start = end = 0;

}

}

cout << "Longest contiguous subsequence: ";

for(size\_t i(max\_start); i <= max\_end; ++i) {

cout << v[i] << ' ';

}

cout << endl;

}

1. You have a class that many libraries depend on. Now you need to modify the class for one application. Which of the following changes require recompiling all libraries before it is safe to build the application?

a. add a constructor

b. add a data member

c. change destructor into virtual

d. add an argument with default value to an existing member function

a. Yes, if default constructor was added

b. No

c. Yes, as distructor signature changes

d. Yes, as function signature changes

1. Implement the following car hierarchy along with the accessor functions and a CarRental class which contains the container to store them.  
   A car rental company wants to keep track of its cars. Each vehicle has a license plate and a brand. (eg. BWM). Currently the company has SUV-s and Sedans. SUV-s have an optional third row seat, sedan’s have an optional sport package. Each car can be queried to inquire the number of passengers it can carry.

class Car {

string plate;

string brand;

public:

Car(const string &plate, const string &brand) : plate(plate), brand(brand) {}

virtual ~Car() {}

const string& GetPlate() const { return plate; }

const string& GetBrand() const { return brand; }

virtual unsigned GetNumberOfSeats() const { return 5; };

virtual void PrintCar() {

cout << ", Plate: " << GetPlate() << ", Brand: " << GetBrand() << ", " << GetNumberOfSeats() << " seats " << endl;

}

};

class Sedan : public Car {

bool sportPackage;

public:

Sedan(const string &plate, const string &brand, bool sportPackage) : Car(plate, brand), sportPackage(sportPackage) {}

bool HasSportPackage() const { return sportPackage; }

void PrintCar() {

cout << "Type: Sedan " << ((HasSportPackage()) ? "with" : "without") << " sport package ";

Car::PrintCar();

}

};

class Suv : public Car {

bool thirdRow;

public:

Suv(const string &plate, const string &brand, bool thirdRow) : Car(plate, brand), thirdRow(thirdRow) {}

bool HasThirdRow() const { return thirdRow; }

unsigned GetNumberOfSeats() const {

return (HasThirdRow()) ? 8 : Car::GetNumberOfSeats();

}

void PrintCar() {

cout << "Type: Suv " << ((HasThirdRow()) ? "with" : "without") << " third row ";

Car::PrintCar();

}

};

class CarRental {

vector<shared\_ptr<Car> > cars;

public:

CarRental(vector<shared\_ptr<Car> > &&cars) : cars(move(cars)) {}

const vector<shared\_ptr<Car> >& getInventory() const { return cars; }

void printCars() {

const vector<shared\_ptr<Car> > &cref = getInventory();

for(size\_t i(0); i < cref.size(); ++i) {

cref[i]->PrintCar();

}

}

};