TDT4102 – Mai 2019, utvidet løsningsforslag (LF) med noen kommentarer.

All kode er testet og kjørt. Vi presenterer som oftest bare én av mange mulige løsninger som ville ha gitt maksimal uttelling på eksamen, men i noen tilfeller flere alternativer. Vi ber om forståelse for at vi ikke har kapasitet til å lage et LF som viser mange ulike løsnings-muligheter på de ulike delspørsmål. (Husk at alle ikke-trivielle programmeringsoppgaver kan løses på utallige måter.) Som tradisjonen er i faget ble det laget ett meget arbeidskrevende oppgavesett, og rettingen viser at den ble litt mer arbeidskrevende enn i fjor. Det er tatt hensyn til dette ved karaktersettingen, og resultatene er omtrent slik de har vært de siste tre årene. Det var veldig mange gode besvarelser, som tyder på at mange har lært mye. Vi jobber nå med forberedelse av kont.eksamen i august.

God sommer!
12/6-2019,
Lasse Natvig

Generell kommentar om rettingen: I de aller fleste av deloppgavene så har en del detaljer slik som linjeskift, komma osv. svært liten betydning for om studenten klarer å vise at han/hun behersker det vi spør om, og da trekkes det ikke for slike detaljer. Vi trekker heller ikke for mindre syntaks-feil som en kompilator raskt ville ha hjulpet deg med.

Oppgave 1: Kodeforståelse

1a) `4 3 4`

1b) `299 -2 1`

1c) `aaabbb 6 ab aCa 2`

1d) `5 5 3`

1e) `7 4`

1f) `dcba bcd`

1g) `2.2 1.1`

1h) `Animal: Jesperpus, Dog: Festus, Dog: Bob, Bob`

1i) `6.6 60`

1j)
```
Pers-copy
Stud-copy
Pers-assign
Stud-assign
```

Oppgave 2:

2a)

```
class Vector2d {
public:
	double x;
	double y;
	Vector2d(double x, double y) : x{ x }, y{ y } { }
};
```

2b)

```
double Vector2d::length() {
	return sqrt(x*x + y*y);
}
```

Kommentar: Vi trakk ikke poeng om funksjonen ble deklarert inline da oppgaven ikke sa noe om det. Bruk av pow(x,2) er like bra som x*x.

2c)

```
Vector2d operator+(const Vector2d &lhs, const Vector2d &rhs) {
	return Vector2d{ lhs.x + rhs.x, lhs.y + rhs.y };
}
```

Kommentar: Vi trakk ikke for manglende const og & siden det ikke sies noe om det i oppgaveteksten

2d)

```
Vector2d Vector2d::operator*(double rhs) const {

   return Vector2d(this->x * rhs, this->y * rhs); // return Vector2d(x*rhs, y*rhs) er også helt OK
}
```

Kommentar: Merk at this ikke er nødvendig, men gjør koden mer eksplisitt (lettere å forstå), vi krevde ikke at de oppdaterer klassen i 2a. Vi trakk ikke for manglende const siden det ikke sies noe om det i oppgaveteksten

2e)

Svar: fordi vi kun har implementert multiplikasjon med Vector2d på **venstre** side av *-tegnet.

2f)

```
ostream& operator<<(ostream& out, const Vector2d& v) {
	return out << "[" << v.x << ", " << v.y << "]";
}
```

Kommentar: Vi trakk ikke for manglende blank bak komma i utskrift

2g)

```
Vector2d vectorSum(const vector<Vector2d>& vectors) {
	Vector2d sum{ 0,0 };
	for (const auto v : vectors) {
		sum = sum + v;
	}
	return sum;
}
```

Kommentar: Trenger ikke bruke auto eller const auto eller for-range for full score.

2h)

```
void trackStats(const vector<Vector2d>& track) {

	double length = 0.0;
	double maxLength = 0.0;
	for (auto segment : track) {
		double len = segment.length();
		length += len;
		maxLength = max(maxLength, len);
	}
	cout << setprecision(3);
	cout << "Length: " << length << " km, ";
	cout << "max-speed: " << round((maxLength * 1000) / 10.0) << " m/min, ";
	cout << "ended at " << vectorSum(track) << endl;
}
```

Kommentar:. Alternativ måte for å få heltall, som static_cast<int>(x) aksepteres

2i)

```
struct Clean {
	Vector2d currentPos;
	Clean() : currentPos{ 0.0, 0.0 } { } // constructor will initialize current pos correctly
	Vector2d operator()(Vector2d delta) {
		Vector2d np = currentPos + delta; // np is short for new position
		if ((np.x < 0.0) || (np.x > 10.0) || (np.y < 0.0) || (np.y > 10.0)) {
			return { 0.0, 0.0 };
		}
		else {
			currentPos = np;
			return delta;
		}
	}
};
void cleanTrack(vector<Vector2d>& track) {
	transform(track.begin(), track.end(), track.begin(), Clean());
}
```

Kommentar: Oppgaven viste seg å bli vanskeligere enn vi hadde forventet, (Stoffet ble forelest ganske grundig, er dekket av eksempelprogram og er godt forklart i boka.) Siden få klarte den ble den gitt redusert vekt. De som klarte å løse oppgaven med andre teknikker enn bruk av transform og funksjonsobjekt fikk opp mot 50% score.

Oppgave 3:

3a)

```
bool operator<(const Food& lhs, const Food& rhs) {

	return (lhs.price < rhs.price);
}
```

3b)

```
void addPrice(map<string, set<Food>>& db, Food fp) {
	db[fp.name].insert(fp);
}
```

3c)

```
void printAllPrices(const map<string, set<Food>>& db) {
	for (const auto what : db) {
		cout << what.first << ":\n";
		for (const auto price : what.second) {
			cout <<  price.price << " " << price.where << endl;
		 }
	}
}
```

Kommentar: - Må ikke bruke const og auto for full score

3d)

```
void bestPrice(const map<string, set<Food>>& db, string name) {
	auto it = db.find(name);
	if (it != db.end()) {
		cout << "Best price for " << name << " is " << it->second.begin()->price
			<< " at " << it->second.begin()->where << endl;
	} else {
		cout << "No price for " << name << endl;
	}
}
```

Oppgave 4:

4a)

```
bool is_inside_rectangle(int x, int y, int r_x, int r_y, int r_width, int r_height) {
	return ( (x > r_x) && (x < r_x + r_width)  &&
		 (y > r_y) && (y < r_y + r_height) );
}
```

Kommentar: Man trenger ikke bruke slike "ekstra" paranteser for å gjøre koden mer eksplisitt for å få full score.
En alternativ løsning kan være:

```
bool is_inside_rectangle(int x, int y, int r_x, int r_y, int r_width, int r_height) {
	if (x <= r_x || x >= r_x + r_width) return false;
	if (y <= r_y || y >= r_y + r_height) return false;
	return true;
}
```

4b)

```
bool is_inside_circle(int x, int y, int c_x, int c_y, int c_rad) {
	int dx = x - c_x;
	int dy = y - c_y;
	return dx * dx + dy * dy < c_rad * c_rad;
}
```

4c)

```
 Color string_to_color(string color) {
	if (colors.find(color) == colors.end())
		throw "Invalid color: " + color;
	return colors.at(color);
}
```

Kommentar: - MERK at vi må bruke .at() siden operator[] krever at Color har en defaultkonstruktør, noe den ikke har. Løsninger som bruker colors[color] ble likevel regnet som fullgode ved sensur. (Dette er mest fordi colors burde hatt dette hvis den var laget litt bedre. (det er en svakhet i graph_lib))
- Merk at koden virker uten de to setningene som er understreket. colors.at() vil kaste en out_of_range unntak hvis fargen ikke finnes, og således er det en god nok løsning med bare den ene kodelinjen.

4d)

```
string color_to_string(Color color) {
	for (auto p : colors) {
		if (p.second == color) {
			return p.first;
		}
	}
	return "unknown color";
}
```

4e)

```
class DAKRectangle : public DAKShape {
protected:
	Rectangle rect;
public:
	DAKRectangle(Point xy, int ww, int hh)
		: DAKShape{ rect }, rect{ xy, ww, hh } { }
};
```

Kommentar:Vi trakk ikke poeng om rect er public, siden det ikke sies ikke noe i oppgaveteksten om rect skal være private, protected eller public. Løsninger med struct ble også godtatt da boka bruker struct like mye som class, og oppgaven ber bare om å bruke arv, men ikke hvordan

4f)

```
string DAKRectangle::to_string() const {
	stringstream ss;
	ss << "rect " << color_to_string(get_color()) << " "
	<< rect.point(0).x << " " << rect.point(0).y << " "
	<< rect.width() << " " << rect.height();
	return ss.str();
}
```

Kommentar: - Trenger ikke bruke stringstream for full score (to_string() og string-konkatenering er helt OK))

```
string DAKRectangle::to_string() {
	Color color = get_color();
	Point xy = rect.point(0);
	stringstream ss;
	ss << "rect ";
	ss << color_to_string(color) << " ";
	ss << xy.x << " " << xy.y << " ";
	ss << rect.width() << " " << rect.height();
	return ss.str();
}
```

4g)

```
 bool DAKRectangle::is_inside(const Point p) {
	Point xy = rect.point(0);
	return is_inside_rectangle(p.x, p.y, xy.x, xy.y, rect.width(), rect.height());
}
```

4h)

```
 void MiniDAK::on_enter_pressed() {
	try {
		do_command(cmd_box.get_string());
		cmd_box.clear_value();
	}
	catch (string err) { // assumes exception thrown as string
		cout << "Error: " << err << endl;
	}
}
```

Kommentar: Vi la ikke vekt på hva slags exception som fanges

4i)

```
 void MiniDAK::on_mouse_click(int x, int y) {
	mouse.x = x;
	mouse.y = y;
	selected_shape = -1;
	for (int i = 0; i < shapes.size(); i++) {
		if (shapes[i].is_inside({ x, y })) {
			selected_shape = i;
		}
	}
}
```

Kommentar: Løsningen ovenfor er enkel og gir full score, En **alternativ løsning**, som har bedre ytelse (men ytelse er noe vi ikke fokuserer på i faget) er å søke bakfra i Shapes og returnere ved første treff:

```
void MiniDAK::on_mouse_click(int x, int y) {
	mouse.x = x;
	mouse.y = y;
	for (int i = shapes.size() - 1; i >= 0; i--) {
		if (shapes[i].is_inside({ x, y })) {
			selected_shape = i;
			return;
		}
	}
	selected_shape = -1;
}
```

4j)

```
void MiniDAK::on_mouse_drag(int x, int y) {
	int dx = x - mouse.x;
	int dy = y - mouse.y;
	if (selected_shape >= 0) {
		shapes[selected_shape].move(dx, dy);
	}
	mouse.x = x;
	mouse.y = y;
}
```

4k)

```
void MiniDAK::save(string filename) {
	ofstream file{ filename };
	if (!file)
		throw "Couldn't open file: " + filename;
	for (auto shape : shapes) {
		file << shape->to_string() << endl;
	}
	file.close();
}
```

4l)

```
void MiniDAK::load(string filename) {
	ifstream file{ filename };
	if (!file)
		throw "Couldn't open file: " + filename;
	string command;
	unsigned int lineno = 1;
	while (getline(file, command)) {
		try {
			do_command(command);
		}
		catch (string e) { // assumes exception thrown as string
			cout << "Error in line " << lineno << ": " << e << endl;
		}
		lineno++;
	}
	file.close();
}
```

Kommentar: Det står ikke spesifisert i oppgaven om load() skal avbryte ved feil eller hoppe over linjen. Begge løsninger er fullgode. Det er minst like bra med løsninger som plasserer try-catch utenfor while-løkken, slik at en stopper lasting av filen ved første feil.

4m)

```
void MiniDAK::do_command(string command) {

	stringstream ss{ command };
	string cmd;
	ss >> cmd;
	if (cmd == "rect") { // Command format: rect color x y w h
		string color;
		int x, y, w, h;
		ss >> color >> x >> y >> w >> h;
		if (ss.fail()) {
			throw string("Invalid arguments");
		}
		DAKRectangle *shape = new DAKRectangle{ { x, y }, w, h };
		shape->set_color(string_to_color(color));
		add_shape(shape);
	}
	else if (cmd == "circle") {…}  // linjen kreves ikke av student
	else if (cmd == "save") { // Command format: save filename
		string filename;
		ss >> filename;
		if (filename.size() == 0) {
			throw string("Missing filename");
		}
		save(filename);
	}
	else if (cmd == "load") {…}// linjen kreves ikke av student
	else {
		throw "Unknown command: " + cmd;
	}
}
```

*Kommentar: B*ruk av filename.length() istf .size() er helt ok. Det som er i LF er nok for å få full score. Det er ikke nødvendig å ha med kode som sjekker om den innleste linjen har for mange argumenter.

...---oooOooo---...