H62SED; Software Eng. Design P Sewell Oct 2013

2013 exam paper solutions

General Comment

Please note that this is a completely open book exam and moreover the students have complete electronic access to all the example programs used to present the course. Therefore, please take into consideration that successful completion of this exam does not involve large quantities of typing, rather judicious “cut, paste and modification”. Of course the test is knowing what to cut and paste and making the design decisions!

The tasks are graded from that required for a basic pass (task 1) through to task 6 which are there to provide scope for 1st class students to do their thing! It is not intended that equal marks are awarded for equal “volumes” of code.

Please note well: I am deliberately testing the student's ability to make engineering decisions as well as their knowledge of *standard good practice* in the subject.

In the marking scheme x+y% for functions means x% for a correct interface and y% for a correct implementation. Just x% requires both correct interface and implementation.

Clearly, there are different equally valid ways of implementing many of the functions and I will use my judgement to deal with any variations within the spirit of the scheme below.

All marks will take account of *suitable* use of in file commentary/documentation

Obviously, the style of the suggested solution is consistent with the level of the module and not necessarily *ideal!*

Task1 (40%) - just for freight_train with no inheritance

Sensible choice of variable names!!!)	5%
Correct use of private	5%
Constructors (initialisation list!)	2+3%
Copy constructor (initialisation list!)	2+3%
Destructor	2+3%
Operator=	5+5%
gets/sets	5%

Task 2 (10%) - just for freight_train with no inheritance

ostream<<	2+3%

ostream>>	2+3%
Task 3 (15%)	5+10%
Task 4 (15%)	
Correct use of pointer array	5%
Dynamic allocation	3%
File access - streams	2%
Implementation (including clarity)	5%
Task 5 (15%)	
public inheritance	3%
correct splitting of member variables	5%
constructors	5%
virtual ~	2%
Task 6 (5%)	
virtual fn - no redecalration for freight_train	5%

```
#include <stdlib.h>
#include <string.h>
#include <iostream>
#include <fstream>

using namespace std;

//-----------------------------------------------------------------------
class train { // Part 5 and 6 only

public:

        train(float _maximum_speed=0):maximum_speed(_maximum_speed){}

        train(const train& t):maximum_speed(t.maximum_speed){}

        train(istream& in){

                in>>*this;
        }

        train& operator=(const train& t){

                if(this==&t) return(*this);

                maximum_speed=t.maximum_speed;

                return(*this);
        }

        virtual ~train(){}

        virtual float get_maximum_speed(){return(maximum_speed);}

        friend ostream& operator<<(ostream& out,const train&);

        friend istream& operator>>(istream& in,train&);

private:

        float maximum_speed;
};
//-----------------------------------------------------------------------
class freight_train:public train { // BASIC PASS IS BLUE

public:

        freight_train(float _maximum_speed,
                    const int _maximum_number_of_cargo_containers,
                    const int _number_of_cargo_containers=0):train(_maximum_speed),

                maximum_number_of_cargo_containers(_maximum_number_of_cargo_containers),
                number_of_cargo_containers(_number_of_cargo_containers){}

        freight_train(const freight_train& ft):train(ft),

                maximum_number_of_cargo_containers(ft.maximum_number_of_cargo_containers),
                number_of_cargo_containers(ft.number_of_cargo_containers){}

        freight_train(istream& in):train(in){
                in>>*this;
        }

        virtual ~freight_train(){}

        const freight_train& operator=(const freight_train& ft){

                if(this==&ft) return(*this);

                train::operator=(*this);

                maximum_number_of_cargo_containers=ft.maximum_number_of_cargo_containers;

                number_of_cargo_containers=ft.number_of_cargo_containers;

                return(*this);
        }

        int get_maximum_number_of_cargo_containers() const {

                return(maximum_number_of_cargo_containers);
        }

        int get_number_of_cargo_containers() const {

                return(number_of_cargo_containers);
        }

        void set_maximum_number_of_cargo_containers(const int _maximum_number_of_cargo_containers) {

                maximum_number_of_cargo_containers=_maximum_number_of_cargo_containers;
        }

        void set_number_of_cargo_containers(const int _number_of_cargo_containers) {

                number_of_cargo_containers=_number_of_cargo_containers;
        }

        int try_to_load_cargo_containers(int no_containers){

                const int available_space(maximum_number_of_cargo_containers-
                                                number_of_cargo_containers);

                const int
no_containers_to_add(available_space>=no_containers?no_containers:available_space);

                number_of_cargo_containers+=no_containers_to_add;

                return(no_containers-no_containers_to_add);
        }

        friend ostream& operator<<(ostream& out,const freight_train&);

        friend istream& operator>>(istream& in,freight_train&);

private:

        int maximum_number_of_cargo_containers;
        int number_of_cargo_containers;
};
//-----------------------------------------------------------------------
ostream& operator<<(ostream& out,const freight_train& ft){

        out<< ft.maximum_number_of_cargo_containers<<" "
          <<ft.number_of_cargo_containers;

        return(out);
}

istream& operator>>(istream& in,freight_train& ft){

        in>>ft.maximum_number_of_cargo_containers>>ft.number_of_cargo_containers;

        return(in);
}
//-----------------------------------------------------------------------
ostream& operator<<(ostream& out,const train& t){

        out<<t.maximum_speed;

        return(out);
}

istream& operator>>(istream& in,train& t){

        in>>t.maximum_speed;

        return(in);
}
//-----------------------------------------------------------------------
int main(){

        int number_of_trains(0);

        ifstream fin("train_file.txt");

        fin>>number_of_trains;

        freight_train** trains=new freight_train*[number_of_trains];

        for(int i=0;i<number_of_trains;++i) trains[i]=new freight_train(fin);

        int number_of_containers(0);

        do {
                cout<<"\nEnter number of cargo containers";

                cin>>number_of_containers;

        } while(number_of_containers<0);

        int next_train(0);

        while(number_of_containers>0&&next_train<number_of_trains) {

                number_of_containers=trains[next_train]-
>try_to_load_cargo_containers(number_of_containers);
```

```
			cout<<"\nTrain number "<<next_train<<" "<<*trains[next_train];

			++next_train;
		}

		cout<<"\n\nThe number of unloaded containers="<<number_of_containers;
}
//-------------------------------------------------------------------------

// train_file.txt

4
100 10 0
100 20 0
100 5 0
100 10 0
```