UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Department of Computer Science
University of Pretoria

Software Modelling
COS121

Examination Opportunity 2 (EO2)

16 September 2015

Name (Initials and Surname): _____

Student number: _____

Examiners

Internal: Dr Linda Marshall, Mr Marius Riekert, Mr Christoph Stallmann and Mr Brian Nyatsine

Instructions

1. Read the question paper carefully and answer all the questions below. / *Lees die vraestel aandagtig deur en beantwoord al die vrae wat volg.*
2. The examination opportunity comprises of **8** questions on **13** pages. / *Die eksamineringsgeleentheid bestaan uit **8** vrae op **13** bladsye.*
3. You have **2** hours to complete the paper. / *Jy het **2** ure om die vraestel te voltooi.*
4. This is an open book paper. You are therefore allowed to have study material with you. / *Hierdie is 'n oopboek vraestel. Jy mag dus studiemateriaal by jou hê.*
5. Please switch off your cell phone, and keep it off for the duration of the paper. / *Skakel asseblief u selfoon af en hou dit af vir die volle duur van die vraestel.*

Question:	1	2	3	4	5	6	7	8	Total
Marks:	5	5	21	12	5	5	10	32	95
Score:	0	4½	14	12	3	5	4	9	47½

Programming Tools / *Programmeerings Gereedskap*

1. Give a Doxygen comment for the function below. You should choose your own description. / *Gee 'n Doxygen komentaar vir die funksie hieronder. Kies jou eie beskrywing.* (5)

```
double calculateAmperage(double watts, double volts);
```

0

2. Your partner in the pair programming practical noticed some strange behaviour with your program. You decide to run **Valgrind** to look for possible problems. / *Jou vriend in die paarprogrammerings prakties het vreemde gedrag in jou program opgemerk. Jy besluit om **Valgrind** te gebruik om vir die moontlike probleme te vind.*

   (a) Name any two types of errors that can be detected using **Valgrind**. / *Noem enige twee tipe foute wat deur **Valgrind** uitgewys kan word.* (2)

   0

   (b) Given your executable name as `prac5`, which command should be used to invoke **Valgrind** on your program? / *Indien die naam van jou uitvoerbare code `prac5` is, watter bevel behoort gebruik te word om **Valgrind** op jou program uit te voer?* (1)

   valgrind prac5 1/2

   (c) After running the above command, you noticed that **Valgrind** doesn't provide enough information to find the mistake. Which flag should be added to the command to provide more detailed information? / *Nadat jy die bogenoemde bevel uitgevoer het, besef jy dat **Valgrind** nie genoeg inligting gegee het om die probleem te uit te wys nie. Watter vlag moet by die bevel bygevoeg word om meer gedetaileerde inligting te kry?* (1)

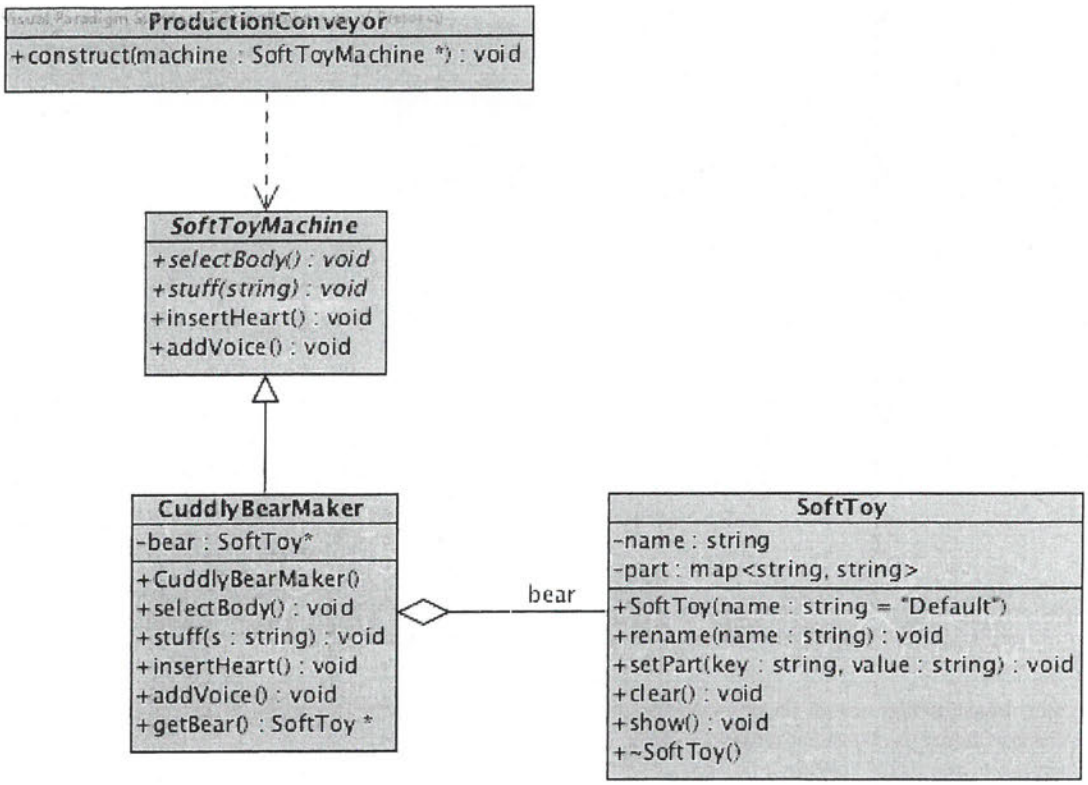
   0

   (d) **Valgrind** shows the following error: `double free or corruption (top): 0x00000000064a7010 ***`. By simply looking at the error message, what did you do wrong? / ***Valgrind** het die volgende fout uitgewys: `double free or corruption (top): 0x00000000064a7010 ***`. Deur net na die foutboodskap te kyk, wat het jy verkeerd gedoen?* (1)

   0

UML / *UML*

3. Consider the following class diagram and answer the questions which follow. / *Beskou die volgende klasdiagram en beantwoord die vrae wat volg.*



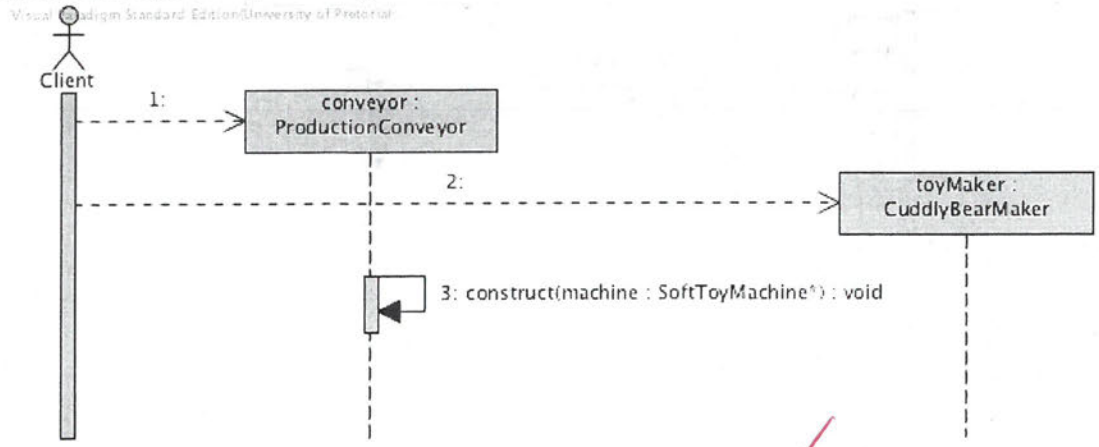
(a) Why is the relationship between `ProductionConveyer` and `SoftToyMachine` a dependency relationship? (1)
/ Hoekom is die verwantskap tussen ***`ProductionConveyer`*** *en* ***`SoftToyMachine`*** *'n afhanklikheidsverwantskap?*

ProductionConveyor uses SoftToyMachine as a parameter in a function.

(b) Assume that the objects `conveyor` and `toyMaker` are defined as follows: */ Veronderstel dat die objekte* ***`conveyor`*** *en* ***`toyMaker`*** *as volg gedefinieer word:* (4)

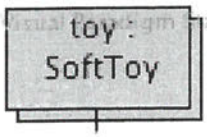
```
SoftToyMachine* toyMaker;
ProductionConveyor*  conveyor;
```

Write down the code represented by the following sequence diagram. */ Skryf die kode neer wat deur die volgende tydsvolgordediagram voorgestel word.*




```
Conveyor = new ProductionConveyor();
toyMaker = new CuddlyBearMaker();
Conveyor -> construct(toyMaker);
```

(c) What does the following lifeline represent and how would you define the object `toy`? / *Wat stel die volgende lewenslyn voor en hoe sal jy die objek* `toy` *definieer?* (2)



only one??

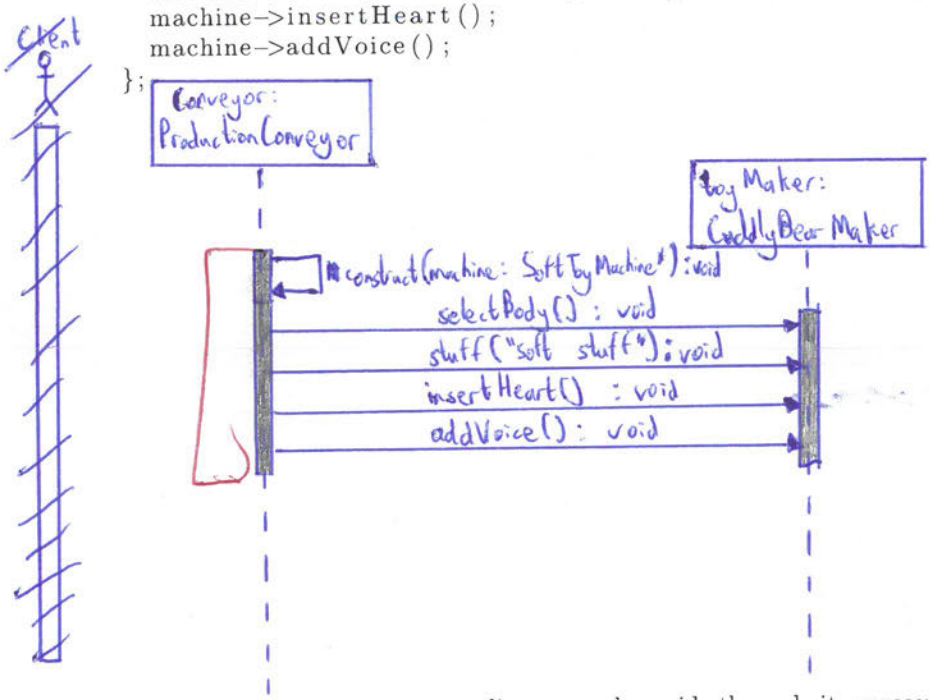
The object is created but never destroyed.

SoftToy *toy = new SoftToy("toy");

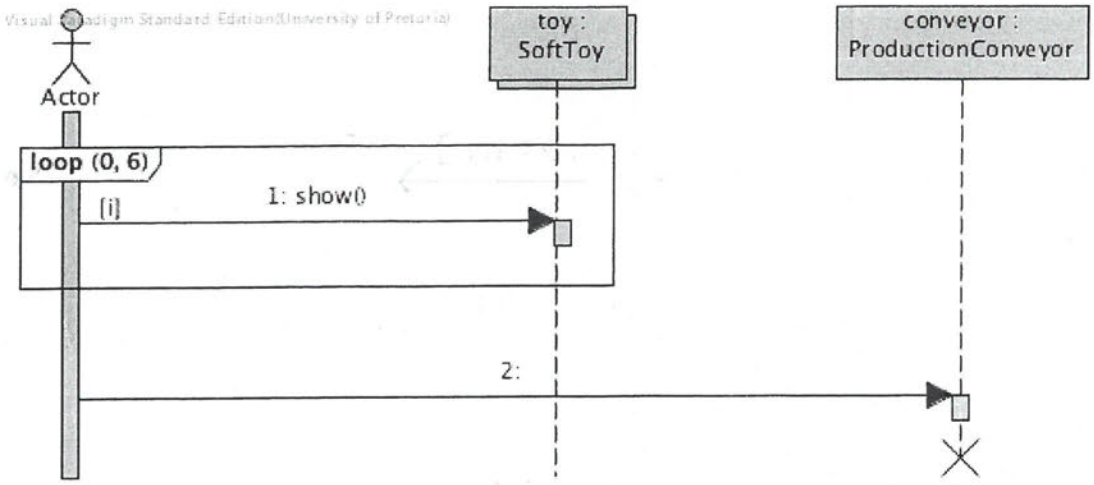
(d) The implementation of the `construct` function defined in `ProductionConveyor` is given below. Draw the sequence diagram, using the objects defined in the previous questions, to show the interaction over time between the objects involved in the interaction. / *Die implementasie van die* `construct` *funksie wat in* `ProductionConveyor` *gedefinieer is word hieronder gegee. Teken 'n tydsvolgordediagram, maak gebruik van die objekte wat in die vorige vrae gedefinieer was, om die interaksie tussen die objekte wat in die interaksie betrokke is oor tyd uit te wys.* (7)

```
void construct(SoftToyMachine* machine) {
  machine->selectBody();
  machine->stuff("soft stuff");
  machine->insertHeart();
  machine->addVoice();
};
```

6



(e) Consider the following sequence diagram and provide the code it represents from the perspective of the main program. / *Beskou die volgende tydsvolgordediagram en verskaf die kode wat dit voorstel vanaf die perspektief van die hoofprogram.* (3)



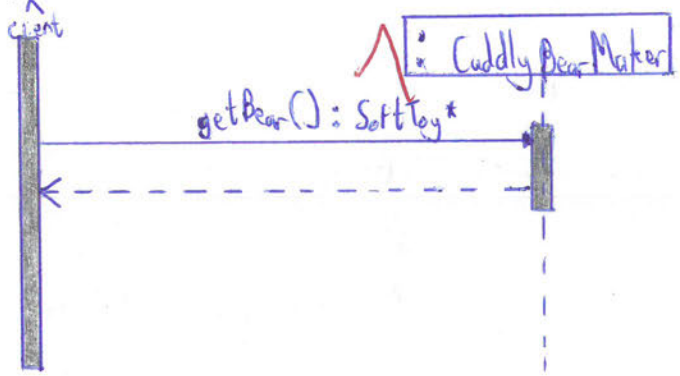
0

```
for (int x=0; x<6; ++x) {
    toy -> show();
~ conveyor;
```

(f) Assuming the `getBear` function is defined as follows. Draw a sequence diagram showing the interaction between the main program and the `CuddlyBearMaker` class when the `getBear` function is called. / *Veronderstel dat die* `getBear` *funksie as volg gedefinieer is. Teken die tydsvolgordediagram wat die interaksie tussen die hoof program en die* `CuddleBearMaker` *klas wanneer die* `getBear` *funksie geroep word wys.* (4)

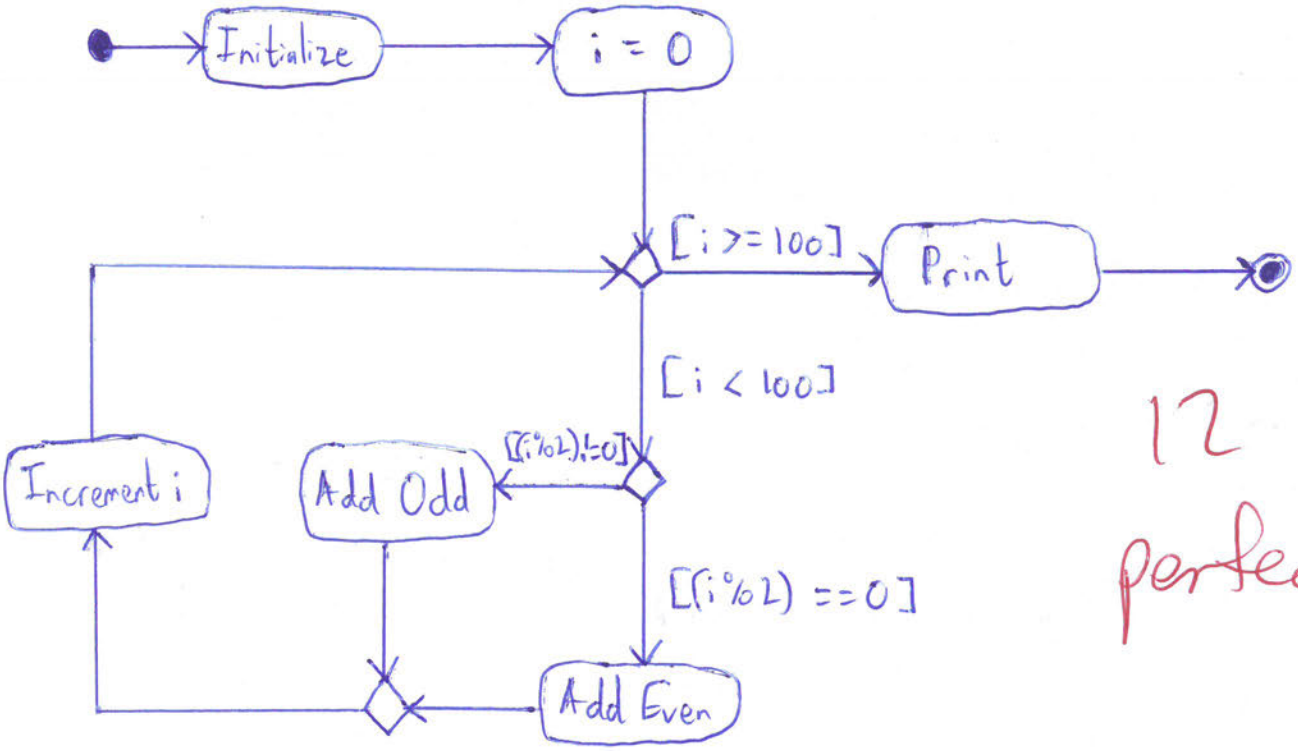
```
SoftToy* CuddlyBearMaker::getBear(){
  return bear;
}
```

3



4. Draw an UML activity diagram from the given code. Both the functions and the activities of `i` should be modelled. Indicate guards where necessary. / *Teken 'n UML aktiwiteitsdiagram van die gegewe kode. Beide die funksies en die aktiwiteite van* `i` *moet gemodeleer word. Dui wagte aan waar nodig.* (12)

```
initialize();
for(int i = 0; i < 100; ++i)
{
        if(i % 2) addOdd();
        else addEven();
}
print();
```



12
perfect!

Design Patterns / *Ontwerpspatrone*

5. What are the benefits of using design patterns when developing software systems? / *Watter voordele is daar verbonde aan die gebruik van ontwerpspatrone in die ontwikkeling van programmatuur stelsels?* (5)

- Help solve problems that occur over and over again
- Address one or more software quality requirements
- Enhances maintainability and adaptibility of code
- Tried and trusted - Improved over decades
- Moves design decisions to a higher level of abstraction.

3

6. Identify the design pattern for each of the descriptions given below. / *Identifiseer die ontwerpspatroon vir elk van die beskrywings wat hieronder gegee word.* (5)

Description / ***Beskrywing***	**Design Pattern** / ***Ontwerpspatroon***
A class that allows you to cycle through a set of objects. / *'n Klas wat jou toelaat om deur 'n versameling objekte te sirkuleer.*	Iterator ✓
If the state of one object changes, the responsibility is dele- gated to a centralised object, which then forwards the notification to all other interested parties. / *As die toestand van een objek verander, dan word die verantwoordelikheid na n sentrale objek gedelegeer wat dan die kennisgewing oordra na alle ander belanghebbende partye.*	~~Observer~~ Mediator ✓
Observe an object and react appropriately when the state of the object changes. / *Neem n objek waar en reageer dienooreenkomstig wanneer die toestand van die objek verander.*	Observer ✓
Allow an object to alter its behaviour when its internal state changes. The object will appear to change its class. / *Laat toe dat 'n objek sy gedrag verander wanneer sy interne toestand verander. Dit lyk asof die objek sy klas verander.*	State ✓
This design pattern is used in situations where it is necessary to dynamically swap out algorithms. / *Die ontwerpspatroon word gebruik in situasies waar dit nodig is om dinamies objekte uit te kan ruil.*	Strategy ✓

5

7. The following illustrates the use of the Observer design pattern in Qt. Study the code carefully. */ Die volgende illustreer die gebruik van die Observer ontwerpspatroon in Qt. Bestudeer die kode noukeurig.*

```
class SecurityManager : public QObject{ /* Implementation */ };
class EmailNotifier : public QObject{ /* Implementation */ };

SecurityManager man;
EmailNotifier email;

QObject::connect(&man, SIGNAL(onBreach()), &email, SLOT(notifyAdmin())); attach
// Some other functionality.
man.onBreach(); // Called when a hacker breaks in. Updat/notify
// Some other functionality.
QObject::disconnect(&man, SIGNAL(onBreach()), &email, SLOT(notifyAdmin())); detache
```

(a) By looking at the given code, what is the intent of the Observer design pattern in the example. */ Deur na die gegewe kode te kyk, bepaal die doel van die Observer ontwerpspatroon in die voorbeeld.* (1)

To observe the QObject class instance(s) and notify/update its dependants.

0

(b) Identify the Observer pattern participants for the code above. */ Identifiseer die deelnemers van die Observer patroon vir die kode soos hierbo gegee.* (8)

Participant or function / *Deelnemer of funksie*	**Class or function in the given code / *Klas of funksie in die gegewe kode***
Observer	QObject ✓
Concrete Observer	SecurityManager ✗
Subject	QObject ✓
Concrete Subject	EmailNotifier ✗
attach()	connect function ✓
detach()	disconnect function ✓
notify()	notifyAdmin function
update()	onBreach function

4

2

(c) Qt made the parent Subject and the parent Observer the same class. What does a shared parent allow the programmer to do that is not do-able in the standard Observer design pattern? */ Qt het die ouer Subject en Observer dieselfde klas gemaak. Wat laat 'n gedeelde ouer die programmeerder toe om te doen, wat nie doenbaar is in die standaard implementasie van die Observer ontwerpspatroon?* (1)

The state does not need to be transfered.

0

Scenarios / *Scenarios*

8. Each of the following questions describes a scenario which can be implemented with using design patterns. For each question you have to identify the most suited design pattern and participants and answer the questions pertaining to each scenario./ *Elk van die volgende vrae beskryf 'n scenario wat geimplementeer kan word met ontwerpspatrone. Vir elk van die vrae moet jy die mees gepaste ontwerpspatroon en deelnemers identifiseer en die vrae antwoord wat betrekking het tot elke scenario.*

(a) Mathematical expressions are usually written in infix notation. It is possible to translate an expression from infix notation to postfix notation. The following is a suggested procedure for translating from infix notation to postfix:

The expression as a string in infix notation is read from left to right one symbol at a time. Two additional strings, one representing operators and one which will eventually become the postfix representation, are initialised as empty strings.

Step 1: If there are unprocessed symbols left for the infix string, obtain the next symbol and go to step 2. Otherwise, move to step 8.

Step 2: If the symbol is a number or a variable, append it to the postfix string and return to step 1. Otherwise proceed to step 3.

Step 3: If the symbol is an operator or a left parenthesis and the operator string is empty or the last symbol in the operator string is a left parenthesis, append the symbol to the operator string and return to step 1. Otherwise, go to step 4.

Step 4: If the symbol is a right parenthesis, start removing symbols one by one from the back of the operator string, appending them in turn to the postfix string until the symbol removed from the operator string is a left parenthesis and return to step 1. Otherwise, move on to step 5.

Step 5: If the symbol is an operator with higher precedence than the last operator appended to the operator string, append the symbol to the operator string and return to step 1. Otherwise, move on to step 6.

Step 6: If the symbol has the same precedence as the last symbol appended to the operator string, and the symbol is left-to-right associative, remove the last symbol from the operator string and append it to the postfix string and append the current symbol to the operator string and return to step 1. Otherwise, if the current symbol is right-to-left associative, simply append it to the operator string and return to step 1. Otherwise, move on to step 7.

Step 7: If the symbol has a lower precedence than the last symbol in the operator string, remove the last symbol from the operator string, append it to the postfix string and return to step 5. Otherwise, move on to step 8.

Step 8: If the operator string still has symbols, remove the symbols one by one back to front from the operator string and append them in this order to the postfix string. Once the operator string is empty, return the postfix string as the final result.

Answer the following:

Wiskundige uitdrukkings word gewoonlik in invoegsel notasie geskryf. Dit is moontlik om 'n uitdrukking van invoegsel notasie na navoegsel notasie om te skakel. Die volgende is 'n voorgestelde prosedure om 'n uitdrukking vanaf invoegsel na navoegsel notasie om te skakel:

Die uitdrukking as 'n string in invoegsel notasie word gelees vanaf links na regs, een simbool op 'n slag. Twee additionele stringe, een wat operators verteenwoordig en een wat uiteindelik die navoegsel notasie sal wees, word as leë stringe geinisialiseer.

Stap 1: *Indien daar onverwerkte simbole in die invoegsel string oor is, kry die volgende simbool en beweeg na stap 2. Andersins, beweeg na stap 8.*

Stap 2: *Indien die simbool 'n getal of 'n veranderlike is, heg dit aan die navoegsel string en keer terug na stap 1. Andersins, beweeg aan na stap 3.*

Stap 3: *Indien die simbool 'n operator of 'n linker hakie is en die operator string is leeg of die laaste simbool in die operator string is 'n linker hakie, heg die simbool aan die operator string en keer terug na stap 1. Andersins, beweeg aan na stap 4.*

Stap 4: *Indien die simbool 'n regter hakie is, begin simbole een vir een verwyder vanaf die agterkant van die operator string en heg hulle om die beurt aan die invoegsel string totdat die simbool wat verwyder is 'n linker hakie is en keer dan terug na stap 1. Andersins, beweeg aan na stap 5.*

Stap 5: *Indien die simbool 'n operator is met 'n hoër voorrang as die laaste operator wat aan die operator string geheg was, heg die simbool aan die operator string en keer terug na stap 1. Andersins, beweeg aan na stap 6.*

Stap 6: *Indien die simbool dieselfde voorrang het as die laaste simbool wat aan die operator string geheg was, en die simbool is links-na-regs assosiatief, verwyder die laaste simbool vanaf die operator string, heg dit aan die navoegsel string en heg dan die huidige simbool aan die operator string en keer terug na stap 1. Andersins, beweeg aan na stap 7.*

Stap 7: *Indien die simbool 'n laer voorrang het as die laaste simbool in die operator string, verwyder die laaste simbool in die operator string, heg dit aan die navoegsel string en keer terug na stap 5. Andersins, beweeg aan na stap 8.*

Stap 8: *As die operator string steeds simbole bevat, verwyder hulle een vir een vanaf die agterkant van die operator string en heg hulle om die beurt aan die navoegsel string. Sodra die operator string leeg is, stuur die navoegsel string terug as die finale resultaat.*

Beantwoord die volgende:

i. Complete the table below by giving the design pattern as well as the role of each of the entities in the pattern:/ *Voltooi die tabel hieronder deur die ontwerpspatroon asook die rolle van elk van die entiteite te gee:* (4)

Entity	Role in chosen pattern
The design pattern	
The party responsible for building the postfix string	
The infix string	
The party responsible for providing the next symbol to be considered from the infix string	
The party responsible for providing the next symbol to be considered from the operator string	

ii. Draw a small UML diagram depicting the design pattern. Do not add any attributes to the classes and substitute participants names with descriptive class names where applicable. / *Teken 'n klein UML diagram om die ontwerpspatroon voor te stel. Moenie enige attribute in die klasse voeg en vervang deelnemer name met geskikte klasname waar van toepassing.* (1)

(b) Assume that a program which will support Boolean algebra is currently under development. The first order of business is to come up with a way to construct and handle Boolean expressions. The following classes have thus far been proposed by the development team, each with a descriptive class name indicating its functionality:

- `Expression`, which is an abstract class.
- `BinaryOperator`, which inherits from `Expression`.
- `Negate`, which inherits from expression.
- `And` and `Or`, which both inherit from BinaryOperator.
- `Variable`, which inherits from Expression and represents a single variable.
- `Constant`, which inherits from Expression.

Answer the following questions:

Aanvaar dat 'n program wat Boolse algebra sal ondersteun huidiglik onder ontwikkeling is. Die eerste orde van besigheid is om op te kom met 'n manier om Boolse uitdrukkings te kan skep en hanteer. Die volgende klasse is tot dusvêr deur die ontwikkelingspan voorgestel, elk met 'n beskrywende klasnaam om die funksionaliteit aan te dui:

- *`Expression`, wat 'n abstrakte klas is.*
- *`BinaryOperator`, wat erf vanaf `Expression`.*
- *`Negate`, wat erf vanaf `Expression`.*
- *`And` and `Or`, wat beide van `BinaryOperator` erf.*
- *`Variable`, wat erf vanaf `Expression` en 'n enkele veranderlike verteenwoordig.*
- *`Constant`, wat erf vanaf `Expression`.*

Beantwoord die volgende:

i. Which class would be most appropriate to encapsulate a single pointer to an `Expression` object?/ *Watter klas sal mees gepas wees om 'n enkele wyser na 'n `Expression` objek te omsluit?* (1)

Variable

ii. Suppose all classes representing binary operations have a single constructor which takes two pointers to `Expression` objects, where the first parameter represents the left hand side of the operator and the second the right hand side. Assume further that the constructor of a class representing a unary operator takes as a parameter a pointer to an Expression object, which is its only operand. The `Variable` class' constructor accepts a single character which represents the name of the variable. Consider the following Boolean expression , where `x` and `y` are variables: (6)

```
(!x && y) || (x && !y)
```

Write a single statement which will create and initialize an object which represents the given expression.
Gestel alle klasse wat binêre bewerkings voorstel het 'n enkele konstrueerder wat twee wysers na `Expression` objekte aanvaar, waar die eerste parameter die linkerkant van die bewerking voorstel en die tweede die regterkant voorstel. Aanvaar vêrder dat die konstrueerder van 'n klas wat 'n unêre bewerking voorstel 'n enkele wyser na 'n `Expression` objek aanvaar wat die bewerking se enigste operand is. Die `Variable` klas se konstrueerder aanvaar 'n enkele karakter wat die naam van die veranderlike voorstel. Aanvaar die volgende Boolse uitdrukking, waar `x` en `y` veranderlikes is:

```
(!x && y) || (x && !y)
```

Skryf 'n enkele stelling neer wat 'n objek sal skep en initialiseer om die gegewe uitdrukking voor te stel.

5

Or resultClass(And(new Negate(new Variable('x')), new Variable('y')),
And(new Variable('x'), new Negate(new Variable('y'))
));

iii. Complete the table below by giving the design pattern as well as the role of each of the entities in the pattern:/ *Voltooi die tabel hieronder deur die ontwerpspatroon asook die rolle van elk van die entiteite te gee:* (6)

Entity	Role in chosen pattern
The design pattern	Template Method
The `And` class	Concrete Class
The `BinaryOperator` class	Concrete Class (Parent + Derived)
The `Variable` class	Concrete Class
The `Expression` class	Abstract Class
The `Constant` class	Concrete Class

0

iv. Draw a small UML diagram depicting the design pattern. Do not add any attributes to the classes and substitute participants names with descriptive class names where applicable. / *Teken 'n klein UML diagram om die ontwerpspatroon voor te stel. Moenie enige attribute in die klasse voeg en en vervang deelnemer name met geskikte klasname waar van toepassing.* (1)

(c) A new strategy game has been released. During development, the developers created two concrete classes representing units, namely `Ornithopter` and `Devastator`. Both of these classes inherit from an abstract class `Unit`. Another class named `ConstructionYard`, which is able to create units of any type upon request, was also created. This class encapsulates objects, one of type `Ornithopter` and one of type `Devastator`. The `Unit` class' definition is as follows:

```
class Unit
{
        public:
                Unit();
                virtual ~Unit();
                virtual int fire() = 0;
                virtual void takeDamage(int) = 0;
                virtual void drawUnit() = 0;

        protected:
                int damage;//The amount of damage a unit can take

};
```

The gaming company has decided to develop an expansion pack. The expansion pack will include a new type of unit called `SiegeTank`. In addition, two upgrades will be available for all units. These are an armor reinforcement, which will increase the amount of a unit, and a rapid fire upgrade, which will allow a unit to fire multiple times in quick succession. These upgrades are "purchased" with the game's currency while playing. Once purchased in a level, the upgrades are applied to all units constructed after the purchase. The upgrades are reset at the start of a subsequent level.

Wrt the expansion pack, answer the following questions:

'n Nuwe strategiespeletjie is vrygestel. Tydens ontwikkeling het die ontwikkelaars twee konkrete klasse geskep, naamlik `Ornithopter` en `Devastator`, om voertuie in die speletjie te verteenwoordig. Beide hierdie klasse erf van 'n abstrakte klas genaamd `Unit`. Daar is ook 'n klas genaamd `ConstructionYard` wat voertuie van enige tipe op aanvraag kan skep. Hierdie klas omsluit twee objekte, een van tipe `Ornithopter` en een van tipe `Devastator`. Die `Unit` klas se klasdefinisie is as volg:

```
class Unit
{
        public:
                Unit();
                virtual ~Unit();
                virtual int fire() = 0;
                virtual void takeDamage(int) = 0;
                virtual void drawUnit() = 0;

        protected:
                int damage;//The amount of damage a unit can take

};
```

Die speletjiesmaatskappy het besluit om 'n uitbreidingspakket te ontwikkel. Die uitbreidingspakket sal 'n nuwe tipe voertuig genaamd `SiegeTank` *insluit. Daarby sal daar twee opgraderings vir alle voertuie beskikbaar wees. Hierdie is 'n wapenuitrusting versterking, wat die hoeveelheid skade wat 'n voertuig kan neem sal verhoog, en 'n versnel skieter, wat 'n voertuig sal toelaat om meervoudige kere opeenvolgend sal laat skiet. Hierdie opgraderings word "gekoop" met die speletjie se geldeenheid terwyl daar gespeel word. Nadat 'n opgradering gekoop is sal dit op alle voertuie toegepas word wat geskep word na die aankoop. Die opgraderings word terug gestel aan die begin van elke opvolgende vlak.*

Mbt die uitbreidingspakket, antwoord die volgende vrae:

i. Write down the class definition for the concrete `SiegeTank` class. Your definition should only have the minimum attributes in order to successfully incorporate it into the game such that it can be treated like any of the other units./ *Skryf die klasdefinisie neer vir die konkrete* `SiegeTank` *klas. Jou definisie moet slegs die minimum atribute bevat sodat die klas suksesvol in die speletjie geintegreer kan word sodanig dit soos alle ander voetuie hanteer kan word.* (2)

```
class SiegeTank : public Unit {
public: SiegeTank();
        virtual int fire();
        virtual void takeDamage(int);
        virtual void drawUnit();
};
```

2

ii. Write down the class definition for a concrete `Upgrade` class. This class should be designed in such a way so that it becomes applicable to any type of unit. Your definition should only have the minimum attributes in order to successfully incorporate it into the game such that it can be treated like any of the other units. All types of upgrades should inherit from this class./ *Skryf die klasdefinisie neer vir 'n konkrete* `Upgrade` *klas. Hierdie klas moet op so 'n manier ontwerp word dat dit van toepassing is op enige tipe voertuig. Jou definisie moet slegs die minimum atribute hê sodat dit suksesvol geinkoroporeer kan word in die speletjie sodat dit soos enige van die ander voertuie hanteer kan word. Alle tipes opgraderings sal erf van hierdie klas.* (4)

```
Class Upgrade {
private:
        Unit * unit;
public:
        Upgrade(Unit*);
};
```

2

iii. Complete the table below by giving the design pattern as well as the role of each of the entities in the pattern:/ *Voltooi die tabel hieronder deur die ontwerpspatroon asook die rolle van elk van die entiteite te gee:* (6)

Entity	Role in chosen pattern
The design pattern	Abstract Factory
The `SiegeTank` class	Concrete Product
The `Unit` class	Abstract Product
A class which will increase a unit's fire rate	Concrete Factory 1
A class which will increase a unit's armour	Concrete Factory 2
The `Upgrade` class	Abstract Factory

iv. Draw a small UML diagram depicting the design pattern. Do not add any attributes to the classes and substitute participants names with descriptive class names where applicable. / *Teken 'n klein UML diagram om die ontwerpspatroon voor te stel. Moenie enige attribute in die klasse voeg en en vervang deelnemer name met geskikte klasname waar van toepassing.* (1)