12 SDD Assessment Task 1- Sarthak Saxena

PART 1-

a) Classify each of the languages used in terms of the computer language generation to which it belongs and justify answers.

-Assembly language- From 2nd Generation. The symbolic representation of machine Instructions, registers and memory addresses allows the programmer to produce a human-readable program. For the computer to understand the program it must be converted to a machine readable format using an Assembler.

-C- Comes from 3rd generations, as it actually contained variables, constants, flow constructs, loops, error handling and etc. Code is translated into machine code for the computer to actually execute.

-Forth- A versatile procedural language originally designed to regulate movement of telescopes and used to control devices and process ranging from heart monitors to special-effects-videos and cameras. From Fourth generation.

b) Why is assemble language described here as rudimentary (elementary or undeveloped) and in this case why would it be used?

-assembly language is described rudimentary because in this case both forth and C are more advanced and assemble language is considered an old generation language comparative to them. It is bad because it takes too long to write a code in assembly, comparative to C and Forth, and is used because programs written in high-level language will usually not run as fast as assembly language programs.

c) Both Forth and C are imperative languages. Compare and contrast imperative and object-oriented languages.

-Imperative languages is where code and data are treated as completely separate, where as in object-oriented the data and code are bundled together into objects. Both are structured well, but the focus of imperative is too use statements to change program’s state. Whereas Object-Oriented is based on the concept of objects which may contain data in the forms of fields, as attributes and code in the form of procedures as methods.

d) Explain the use of the three different languages in this scenario rather than a single all-purpose language.

- As all languages have a different purpose as assembly can’t always be used as it is great on oneside of coding but takes way too long to finish, and function correctly. As well as the other languages make it easier for programmers to code, as the language is more code friendly and developed to add a better structured, detailed code. Rather than a mess of just one language, which will be chaos to figure out where the problems are, for a certain program which does not support that language.

Question 2

a) Describe all possible results from running the code on line 22.

The results are – go:- parent(X, Melody), then it would run parent(X,Y) if father (X,Melody). Then it would return with parent (Jim,Melody) if father (Jim,Melody). Therefore making Jim parent of Melody. Then it would also find running Parent (X,Y) if mother (X,Melody), and will find that mother( Eleanor, Melody) Therefore parent (Eleanor, Melody) is true and Eleanor and Jim are parents.

b) The writer of the code wishes to improve it so that it includes a definition of a sibling. A sibling is simply someone who is either someone’s brother or sister. Using the code given as examples of syntax, extend the program so that it includes this definition. Your answer should be concise as possible.

The way to add the rule of sibling it will add a test like this

**sibling**(X, Y) :- mother(M, X), father(F, X), mother(M, Y), father(F, Y),X\= Y.

which will call all people and see if they are siblings, by assigning them to X, and then testing to see if they have the same mother and father, from mother(M, X) and father(F, X). Then it would also call ‘Y’ to see if the father and mother of Y is the same as X. Then it would return with all siblings like’

X = Crystal

From sibling(X, Melody).

Question 3-

class Person

{

private:

Int phonenumber;

Public:

Int getphonenumber()

{

return phonenumber;

}

};

The attribute is phone number, and the method basically gets the phone number and then returns it.

[Part 2] Question 4-

#include <iostream>

#include <string>

class Hero //Base class

{

private:

int lives;

int health;

int mana;

public:

int getlives()//gets lives for the next part

{

return lives;

}

void setlives(int i)//sets lives for when the Hero lives is instantiated.

{

lives=i;

}

int gethealth()

{

return health;

}

void sethealth(int h)

{

health=h;

}

int getmana()

{

return mana;

}

void setmana(int m)

{

mana=m;

}

};

class Enemy

{

private:

int lives; //cannot be accessed by Minions as it is privated.

public:

int health;

int damage;

int getlives()

{

return lives;

}

void setlives(int i) //example of polymorphism as this overwrites lives from hero to Enemy.

{

lives=i;

}

int gethealth()

{

return health;

}

void sethealth(int h)

{

health=h;

}

int getdamage()

{

return damage;

}

void setdamage(int d)

{

damage=d;

}

};

class Minions: public Enemy //example of inheritance as minions inherit stats of Enemy

{

int getminionhealth()

{

return health;

}

void setminionhealth(int mh)

{

health=mh;

}

int getminiondamage()

{

return damage;

}

void setminiondamage(int md)

{

damage=md;

}

};

int main()

{

Hero A; //intializes code for Hero

std::cout<< "-----------------------WELCOME TO WROTHGAR----------------------------" << "\n" << "----------------------THE LEGENDARY FIGHTER---------------------------" << "\n";

char name[50];

std::cout << "Generic Hero:";

std::cin >> name;

std::cout << "You are " << name << " of Wrothgar" << "\n";

A.setlives(3);// initalizes lives of hero to 3.

std::cout<<"You have: " << A.getlives()<< " lives" << "\n"; //output will be 3

A.sethealth(10); //intitalizes health of hero to 10.

std::cout<< "Health: " << A.gethealth() << "\n";

A.setmana(20); //initializes mana of hero to 20.

std::cout<< "Mana: " << A.getmana() << "\n";

std::cout<< "----------------------------------------------------------------------";

Enemy B;

std::cout << "\n" << "Name your Enemy:";

std::cin >> name;

std::cout << "You are challenged by " << name << " of Kortana" << "\n";

B.setlives(1); //initializes lives of enemy to 1

std::cout<<"He has: " << B.getlives()<< " lives" << "\n";

B.setdamage(2);

std::cout<<"Look out he does: " << B.getdamage()<< " hearts of damage" << "\n";

Minions C;

std::cout<<"Oh no " << name << " has minions" << "\n";

std::cout<< "Grumpy-(>.<)" << " Bumpy- (~.~)" << "\n"; //displays Minions.

std::cout<< " /(# #)/" << " /(# #)/" << "\n";

std::cout<<" ! !" << " ! !" << "\n";

C.sethealth(1);//initializes minion health to 1

std::cout<< "Minion Health: " << C.gethealth() << "\n";//output will be 1

C.setdamage(1);//initializes the minion damage to 1

std::cout<< "Minion damage: " << C.getdamage() << "\n";//output of damage: 1.

};

Question 5:

The program has void setlives(int i) and right after that has  
void setlives(int i) which displays polymorphism as the code is being overwrited from hero lives to Enemy lives.

The inheritance is displayed when class Minions: public Enemy.  
To show that all the traits that enemy had public have been inherited by Minions.

Encapsulation is displayed when the lives are being privated from minions, in that inherited class. As when you try to access lives in minions it gives an error because of the lives being private.

Abstraction is displayed throughout the code as the base class has many things being defined, as well as methods, and objects.

Question 6

**dinosaur**(tyrannosaurus\_rex). % facts or the inference engine as this is what the main info is stored.

**animal**(lion).

**animal**(eagle).

**animal**(rabbit).

**has\_whiskers**(lion).

**has\_feathers**(eagle).

**has\_beak**(eagle).

**prey**(tyrannosaurus\_rex, lion). % get's taken when finding out predator

**prey**(lion, eagle).

**prey**(eagle, rabbit).

predator(X, Y) :- prey(X, Y). %Goals %backwards or forwards chaining can be used to find the predator of X and Y.

bird(X) :- %Rules %Classifies as an example of Heuristics

animal(X),

has\_feathers(X),

has\_beak(X).

cat(X) :- %states what is required for something to classify as a cat.

animal(X),

has\_whiskers(X).

extinct(X) :-

animal(X),

dinosaur(X).

Question 7  
Heuristics is displayed in lines 1-11 as the tests carried out give an answer of false, because the overall answer is false for all the animals being predator of a certain animal. Still gives the right answer but then redoes the query and tests for the other animals and returns as an overall false, therefore deeming this code to have an example of heuristics.

Inference engine is displayed in lines 1-10 as the data of information is stored as facts in those lines, basically identifying them as an animal, dinosaur, has whiskers, feathers, beak and prey.

Backward chaining is used to find out if predator of rabbit is eagle, as it first checks if rabbit is prey which means it will check if it is an animal, and then it’ll come true, and then check if it kills an animal, or gets killed by an animal. It will find that eagle kills rabbit, through testing all animals, and if it says prey(eagle, rabbit) and then come back true. Which is an example of Backward chaining going through the code to find out if eagle is the predator of rabbit.

Peer Code Review- Question 8 [Part 8]

Sarthak’s code:-

Need to add-

* Sections of weapons, and classes that correspond to them.
* Male or Female
* A welcome statement, and then a continue
* Add defense as an attribute
* Add comments to better guide
* Add an example of polymorphism
* A derived class to state warrior, barbarian and archer
* Control structures like if statements