24) a) Fragment 1 - logic or declarative as it declares a set of rules and facts and then queries it using a ?- "statement" this is hybrid prolog syntax and invokes forward or ~~backward~~ backward chaining using a set of rules + facts to reach a conclusion or resolution as to whether the goal is true or false or to prove the validity of a statement.

Fragment 2 → ~~Procedural~~, Functional as it involves the use of many brackets and data values. (lists and atoms) and it can ~~use~~ use recursion to process lists of data. This are based on mathematical functions.

BOARD OF STUDIES
NEW SOUTH WALES

Centre Number: ..... Student Number: .....

Question 24.

b) The object-orientated paradigm arose because of the ~~[illegible]~~ emergence of more sophisticated hardware.

Continued next page

BOARD OF STUDIES
NEW SOUTH WALES

Centre Number: ..... Student Number: .....

Programming languages could be created that utilised individual objects with their own data & methods because the technology was now available to use this. There was also a desire for greater productivity. Object-orientated programming mimics the way we think and thus programmers are able to understand & program efficiently with it without getting bogged down in the syntax of the language. The approach to testing in object orientated programming means each object can be tested individually, then in combination with other objects, speeding up the coding process.

There was also a recognition of standard tasks written by programmers which could be utilised in OO programming by re-using modules of code.

BOARD OF STUDIES
NEW SOUTH WALES

Centre Number: ..... Student Number: .....

There was also a desire to solve different types of problems which O-O programming allows. The learning curve is also reduced by O-O languages, which was fuelled by increased end-user programming.

The advent of the GUI also fuelled the emergence of O-O languages because this paradigm is suitable for event driven programs which is an element of the GUI.

The object-orientated paradigm also arose from the acknowledgement that many 'objects' could have similar characteristics to speed up code generation, hence polymorphism & inheritance.

(c)

(i) The problem is that when the program is run, InstRectangle.height equals zero, (line 20) so the condition of the while loop is false, meaning the contents aren't executed.
One solution would be to use a post-test loop.
Another would be to give InstRectangle.height a non-zero value when the program is first run (before line 20)

(ii) type

```
PTriangle = ^TTriangle;
TTriangle = object (TObject)
private
```

PTO.

```
        base, height : integer;
    public
        function area : integer;
    end;
function TTriangle.area : integer;
begin
    area := ((1/2) * base * height);
end;

var
    InstTriangle : TTriangle;
```

(d) I would choose the logic paradigm.
The logic paradigm is best when decisions have to be made by the system by itself, based on previously defined rules - such as the 'compression rules' in this situation.
Also, the logic paradigm was designed specifically for artificial intelligence and robotics - which this system utilises.

At first, I considered using the object-oriented paradigm, with 'bag' and 'chute' objects, but the automated decisions to be made (based on rules - and inferences) suggest that the logic paradigm would be more suitable; as object-oriented programming offers no easy way of making 'intelligent' decisions.

The functional paradigm is ~~totally~~ inappropriate, because it is useful primarily for mathematical processes. Also, programs written in functional languages are often ~~difficult~~ executed as they are written, making them unsuitable for permanent unmanned operations. Functional programs are also difficult to read, and therefore maintain, because of their nested functions.