BOARD OF STUDIES
NEW SOUTH WALES

Question 24

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

a) Fragment 1 ~~uses~~ has been coded using the logic paradigm. This is evident because of the use of a set of ~~rules~~ facts ~~e.g.~~ to e.g hair(sally, red) and then a query e.g ?- hair(sally, X). Fragment 2 is of the functional paradigm. The features of the paradigm present are functions and recursion.

b) The object orientated paradigm has emerged as a result of the development of Graphical User Interfaces (GUI's) and the need for greater productivity and ease of use of software. ~~The imperative paradigm~~ With the emergence of GUI's, tasks no longer needed to be executed in a set order. Users were able to select from many different tasks on screen so it became more efficient for an object orientated paradigm to emerge. Code was now activated when the user invoked it through objects on the screen. ~~and thus~~ It also ~~came~~ emerged through the increase in demand for highly interactive software ~~and~~ games, multimedia that could not follow ~~as~~ a set ~~path~~ sequence such as in the imperitive paradigm.

Question 24

BOARD OF STUDIES
NEW SOUTH WALES

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

The object orientated paradigm emerged also from the need for greater productivity as well tested and secure objects could simply be put together to synthesise a much larger program. This meant for less development time and easier maintenance of code.

c) i) The input for the rectangle height is located within the loop that uses it as a condition. This prevents the program from running. To fix this you could move the input to before the loop. Alternatively you could change the repetition to a post-test (or unguarded loop) either method would solve the problem.

Question 24

BOARD OF STUDIES
NEW SOUTH WALES

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

ii)

```
type
  PTriangle = ^TTriangle
  TTriangle = object(TObject)
  private
    height, base : integer;
  public
    function area:integer;
  end;
function TTriangle.area:integer;
begin
  area := (height * width);
end;
```

d) The logic or declarative paradigm is best suited to this problem. Once the barcode has been read on the bag the facts can be passed through a set of rules which will determine which chute the bag will be placed in. e.g. Rules:- chute1(Firstclass, Sydney)

chute2(Firstclass, Melbourne)

chute3(Economy, Sydney)

and the query would be structured:

?(Class, Destination)

Informing the system of which chute to place the bag in.

These rules are extracted from the flight schedule each day. 'Compression rules' will show where the number of active chutes can be reduced by adding to the rules. Eg.

Flight (Melbourne, Stopover)

Flight (Sydney, Terminate)

Flight (Adelaide, Destination)

If a flight is merely stopping over or refuelling then the baggage need not be seperated by destination in that case.