



Question 2f.

a).

FRAGMENT 1: Logic paradigm; because facts are stated such as `hair(sally, red)` which means sally has red hair. ~~.....~~

A query is also included `?-hair(sally, X)` which is querying the colour of sally's hair. Appears to be PROLOG code.

FRAGMENT 2: Functional Paradigm. The code is made up ~~of~~ entirely of functions. The brackets indicate functions & the code shows smaller functions combining to make larger functions indicated by the outer brackets. The command `setg` is an element of LISP which is a functional language.

b) Object oriented languages have increased productivity greatly due to, primarily, the use of re-usable code ~~and~~ and routines from run-time-libraries. This quickens the development process and reduces testing time as the data routines and modules do not need to be ^{rigorously} tested as they are known to work.

OOP (Object oriented programming) allow for software modelling abilities, as objects can model real-world - functions and features being particularly useful for simulations and games.

OOP also allow encapsulation, where data and methods are encoded up in an object, making for secure systems. →

OOP also allows for independent compilation, where each module is tested independently before being integrated, allowing for greater error detection and isolation.

OOP has also allowed for GUI applications, with objects being widgets on the screen.

Also inheritance has allowed for efficiency and ~~to consistency~~ consistency of objects, ~~and~~ classes and subclasses, which allow for a better system design.

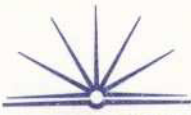
This OOP has and with continue to increase efficiency and productivity of coding.



c) The rectangle height is not read before ~~the relative operation~~. $\text{Inst.Rectangle.height} < 0$ is used to terminate the loop \therefore the loop will never execute, ~~if~~ as $\text{Inst.Rectangle.height}$ is 0 to start with.

This ^{logic error} ~~solution~~ could be solved by:

1. Placing a ~~readln~~ $\text{readln}(\text{Inst.Rectangle.height})$ before the while loop.
- ~~2. Changing~~ to read an initial value for height.
2. Changing the while (pre-test loop) to a repeat (post-test loop) with a terminating



Condition of:
until InstRectangle.height = 0

ii) type

```
PTriangle = ^T Triangle;  
TTriangle = object (T object)
```

```
private
```

```
base, height: integer;
```

```
public
```

```
function area: int real;
```

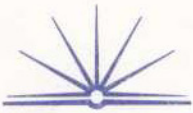
```
end;
```

```
function TTriangle.area.real;
```

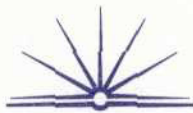
```
begin
```

```
area := ((base * height) / 2)
```

```
end;
```



A logic paradigm would be used to develop the system. The proposed system demands a level of logic from the program. For the program to determine where bottlenecks may occur or when less choices can be used, a logic paradigm allows the definition of facts and rules, which can be put ~~together~~ together to reach a goal, by heuristics. In



This way, the system will be able to cope intuitively with any of the complexities that may arise, using the 'Compression Rules' that are ~~are~~ decreed as needing to be used. A logic paradigm is well suited to this task.

If the facts on passenger bookings are updated regularly throughout the day, the program will be able to account for any fluctuations in passenger ~~the~~ bookings, and adjust the charters accordingly.

The fact that the path by which the program follows to solve the problem does not need to be known, the system just needs to work reliably, quickly, and efficiently. This is due to the logic paradigm. Using heuristics will reach a correct goal without necessarily ~~not~~ being able to show how the decision was reached or whether it is the absolutely correct solution. Nevertheless, the system, created using a logic paradigm, will reach a goal based on its given facts, and so is a good choice for the development of the ~~system~~ system.