Question 1 – Invariants (8%):

Thirteen large empty boxes are placed on a table. An unknown number of the boxes are selected and, into each, six medium boxes are placed. An unknown number of the medium boxes are selected and, into each, six small boxes are placed.

At the end of this process there are 63 empty boxes on the table. **How many boxes are there in total?** You should use the following steps in order to solve the problem. The variables *e* and *t* represent the number of empty and the number of total boxes, respectively.

1. Identify the information that is given about the initial and final values of *e* and *t*. (2%)
2. Model the process of putting six boxes inside a box as an assignment to *e* and *t*. (2%)
3. Identify an invariant of the assignment (proof required!). (2%)
4. Combine the previous steps to deduce the final value of *e.* (2%)

Answer :

Initially : $t = 13$, $e = 13$; Finally : $t = ?$, $e = 63$

When placing 6 medium boxes into large box , it starts with 1 empty large box and will ends up with 6 empty medium boxes and 1 large box, which is 7 total boxes and 6 empty boxes. Therefor : $e , t := e + 5 , t + 6$

Set invariant is : $xe + yt$

$(xe + yt)[e , t := e + 5 , t + 6]$
$= (xe + 5x) + (yt + 6y)$
$= (xe + yt) + (5x + 6y)$

$5x + 6y = 0$ is true when $x = 6$, $y = -5$, so invariant is : $6e – 5t$

Initial value of invariant : $6 * 13 – 5 * 13 = 13$

Final value of invariant : $6 * 63 – 5t = 378 – 5t$

$378 – 5t = 13$, $5t = 365$, $t = 73$

Question 2 – Calculating Programs (12%)

A team of programmers are developing software to assist in solving a range of mathematical problems. A component of the software is based around the following code snippet:

```
{Precondition: n > 0}

result := 1
for (i = 2; i < n + 1; i++)
   result := result * i

{Postcondition: result = n!}
```

The loop is designed to operate from states satisfying the precondition $\{n > 0\}$ and

end in states satisfying the postcondition $\{result = n!\}$.

The team wants to ensure that their software operates correctly, and request that you verify the loop using induction. You must explicitly identify the base case and induction hypothesis.

You should use the following steps in order to solve the problem:

1. State explicitly the base case of the program and induction hypothesis (2%)
2. Show how to use the induction hypothesis to solve the inductive step (10%)

Answer :

Base case n = 1

```
{1 > 0}

result := 1
for (i = 2; i < 2; i++)
   result := result * i

{result = 1! = 1}
```

Loop doesn't run for the base case , so

```
{1 > 0}

result := 1

{result = 1! = 1}
```

∴ Hoare triple is valid

The induction hypothesis is for any problem of size n, the solution to a problem of size n + 1 will be identical to the solution for the problem of size n.

Assume validity for n:(n > 0) , replacing n with n + 1 to prove validity for n + 1:

```
{n > 0}

result := 1
for (i = 2; i < n + 1; i++)
   result := result * i
{result = n!}

result := result * (n + 1)

{result = (n + 1)!}
```

Reduced to

```
{result = n!}

result := result * (n + 1)

{result = (n + 1)!}
```

Therefor the assignment is

```
result = n! => (result = (n + 1)!)[result := result * (n + 1)]
```

Solving by RHS assignment rule

```
(result = (n + 1)!)[result := result * (n + 1)]
```
{substitution}
```
result * (n + 1) = (n + 1)!
```
{assume `result = n!`}
```
n! * (n + 1) = (n + 1)!
```
{simplify}
```
n! * (n + 1) = n! * (n + 1)
```

∴ Hoare triple is valid