Q1: 1. Check that Request <= Need (that is, (2,1,0) <= (0,1,1) = >False

<> request cannot be granted. There is erorr

Q2: 1. Check that Request <= Need (that is, (4,3,3) <= (6,0,0) = >False

<> request cannot be granted. There is erorr

Q3: 1. Check that Request <= Need (that is, (1,2,0) <= (4,3,1) = >true

2. Check that Request <= Available (that is, (1,2,0) <= (3,3,2) => true

***Available* = *Available* – *Requesti;***

***Allocationi*= *Allocationi* + *Requesti*;**

***Needi* = *Needi* – *Requesti;***

*Allocation Max Available Need*

*A B C A B C A B C A B C*

*P*0 0 1 0 7 5 3 3 3 2 7 4 3

*P*1 2 0 0 3 2 2 1 2 2

2 1 2

4 2 3

6 2 3

9 2 5

10 4 7

10 5 7

*P*2 3 0 2 9 0 2 6 0 0

*P*3 2 1 1 2 2 2 0 1 1

4 3 1-

1 2 0=

3 1 1

→ *P*4 1 2 2 4 3 3 3 1 1←

0 0 2 +

1 2 0 =

1 2 2

Executing safety algorithm shows that sequence <P3,P1,P2,P4,P0>satisfies safety requirement so request can be granted.

Q4: 1. Check that Request <= Need (that is, (3,3,0) <= (7,4,3) = >true

2. Check that Request <= Available (that is, (3,3,0) <= (3,3,2) => true

***Available* = *Available* – *Requesti;***

***Allocationi*= *Allocationi* + *Requesti*;**

***Needi* = *Needi* – *Requesti;***

*Allocation Max Available Need*

*A B C A B C A B C A B C*

0 1 0 +

3 3 0 =

3 4 0

→ *P*0 3 4 0 7 5 3 3 3 2 4 1 3←

7 4 3-

3 3 0=

4 1 3

*P*1 2 0 0 3 2 2 1 2 2

0 0 2

*P*2 3 0 2 9 0 2 6 0 0

*P*3 2 1 1 2 2 2 0 1 1

*P*4 0 0 2 4 3 3 4 3 1

<> request cannot be granted.