Question A.

1.

- Given block size B 2048 bytes
- The record of the size R can be calculated as:
  $R = 10 + 4 + 5 + 12 + 12 + 15 + 4 = 62$
- Therefore, bfr can be calculated as:

$$\mathrm{bfr} = \left\lfloor \frac{2048}{62} \right\rfloor \approx 33$$

- As a result, the number of file blocks needed to store the EMPLOYEE records is:

$$b = \lceil \frac{12000}{33} \rceil = 364 \text{ blocks}$$

2.

- Given the key field size of 15 bytes and the block pointer size of 6 bytes
- The index entry size $R_1 = 15 + 6 = 21$ bytes
- The index blocking factor $Bfr_1 = \frac{2048}{21} \approx 97$
- The number of index blocks:

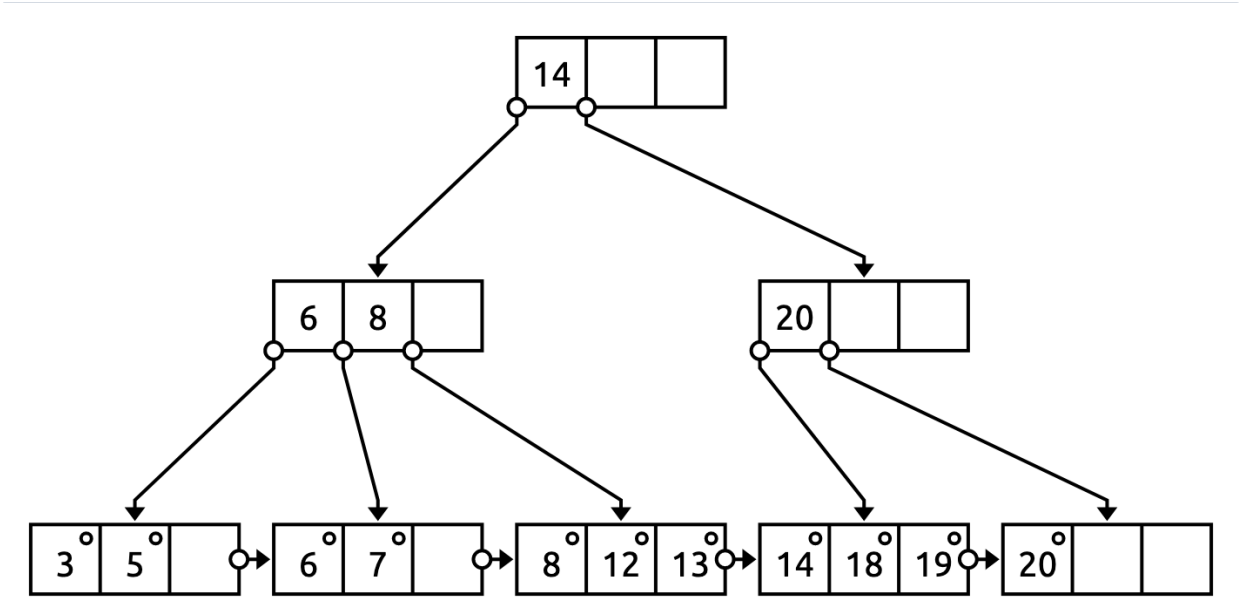
$$b_1 = \lceil \frac{364}{97} \rceil = 4$$

3.

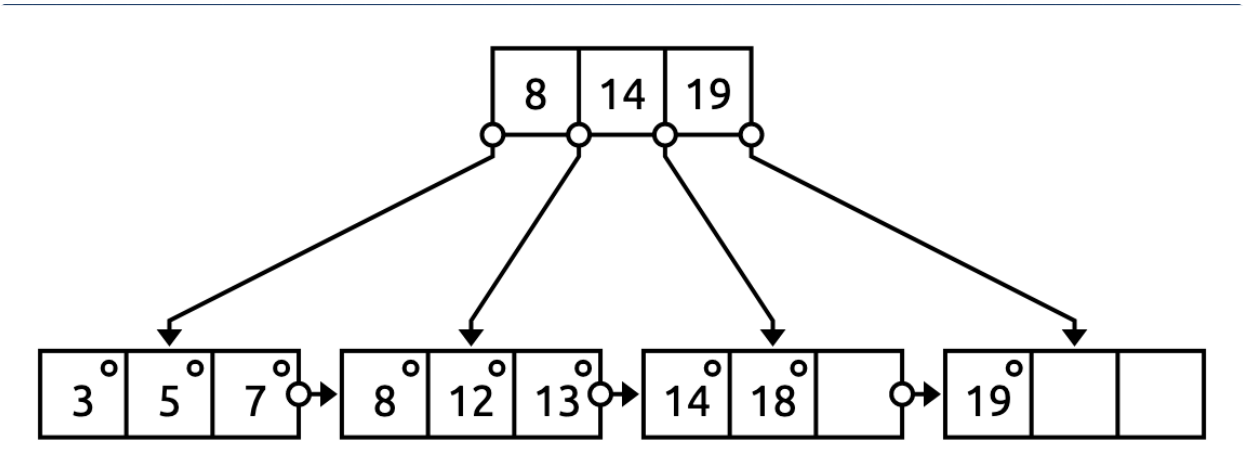
- $Bfr_I = 97$
- $b_1 = 4$
- $b_2 = \frac{4}{97} \approx 1$
- Therefore, the total number of occupied blocks used for the index is $4 + 1 = 5$ blocks.

Question B.

1.



2.



Question C.

Schedule 1

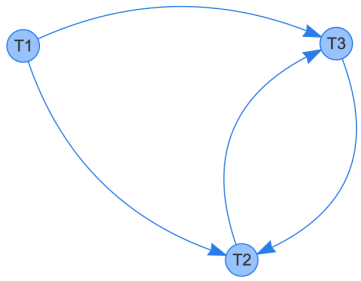
T_1	T_2	T_3
$r_1(X)$		
$w_1(Y)$		
	$r_2(Y)$	
$r_1(Z)$		
		$w_3(Z)$
		$r_3(X)$
		$r_3(Y)$
	$w_2(X)$	
c_1		
	$r_2(Z)$	
		$w_3(X)$
	$w_2(Y)$	
		c_3
	c_2	

Schedule 2

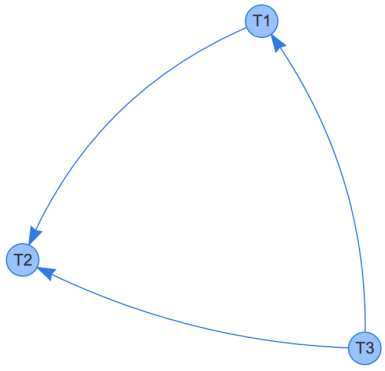
T_1	T_2	T_3
		$w_3(Z)$
		$r_3(X)$
		$r_3(Y)$
		$w_3(X)$
		c_3
$r_1(X)$		
$w_1(Y)$		
$r_1(Z)$		
c_1		
	$r_2(Y)$	
	$w_2(X)$	
	$r_2(Z)$	
	$w_2(Y)$	
	c_2	

1.

Schedule 1 SG



Schedule 2 SG

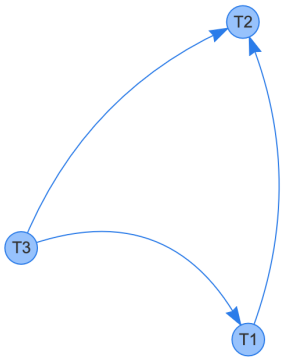


- As a result, Schedule 1 is not serialisable due to a cycle, while Schedule 2 is considered serialisable because it is acyclic.

Equivalent Serial Schedule 2

T_1	T_2	T_3
		$w_3(Z)$
		$r_3(X)$
		$r_3(Y)$
		$w_3(X)$
		c_3
$w_1(Y)$		
$r_1(X)$		
$r_1(Z)$		
c_1		
	$w_2(X)$	
	$r_2(Y)$	
	$r_2(Z)$	
	$w_2(Y)$	
	c_2	

Equivalent Serial Schedule 2 SG



2.

Schedule 1 is recoverable, non-strict and cascaded. It is because T_2 and T_3 read a data item Y written by T_1, and T_1 committed before T_2 and T_3. Also, T_2 read a data item Z written by T_3, and T_3 committed before T_2. Therefore it is recoverable. However, T_2 read a data item Y written by T_1, and T_1 has not yet committed. Therefore, it is non-strict and cascaded.

Schedule 2 is obviously strict, which also means it is recoverable and cascadeless. This is because a transaction can neither read nor write a data item until the last transaction that wrote the data item has been committed.