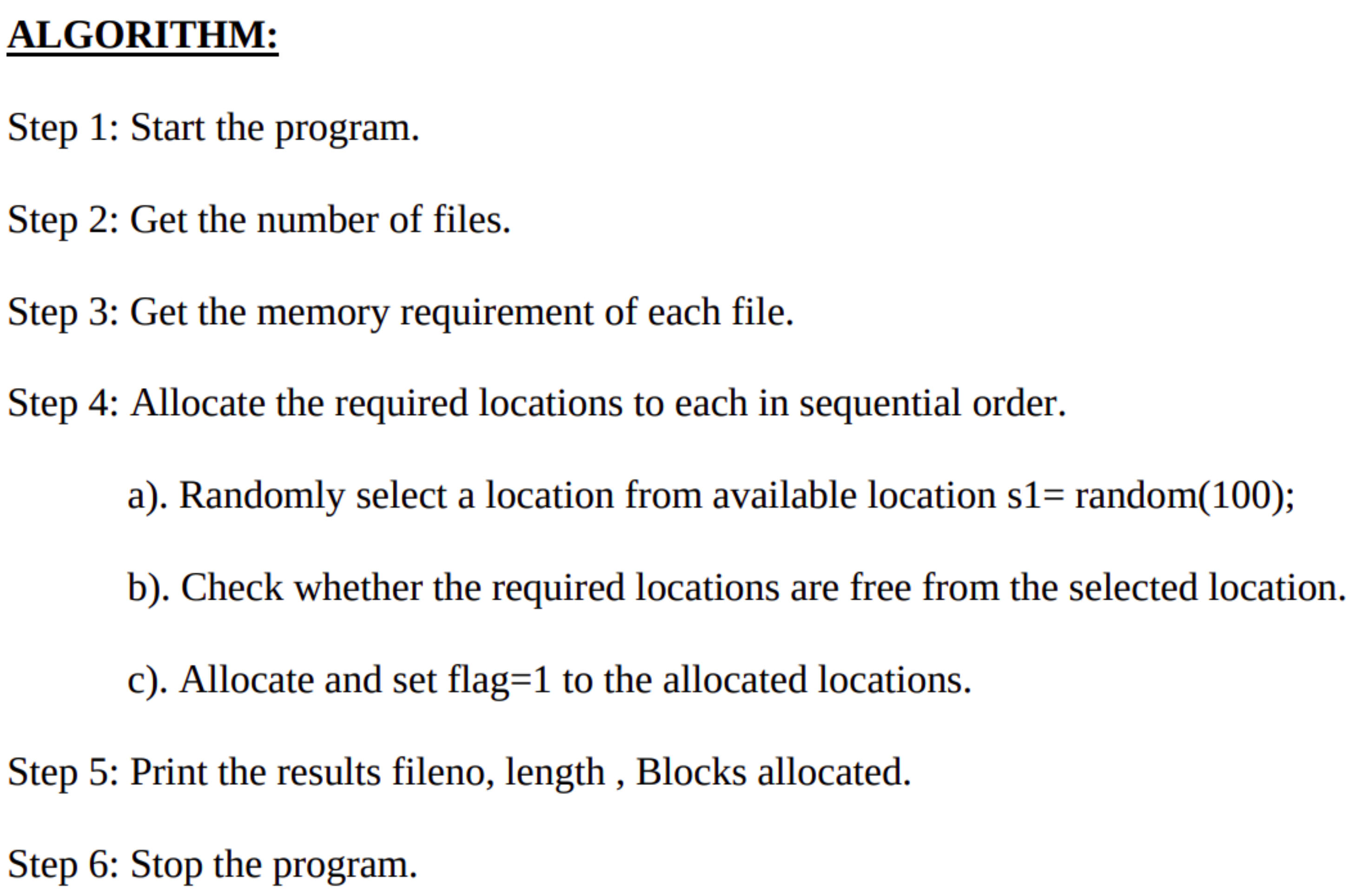
Sequential Allocation



ALGORITHM:

Step 1: Start the program.

Step 2: Get the number of files.

Step 3: Get the memory requirement of each file.

Step 4: Allocate the required locations to each in sequential order.

a). Randomly select a location from available location s1= random(100);

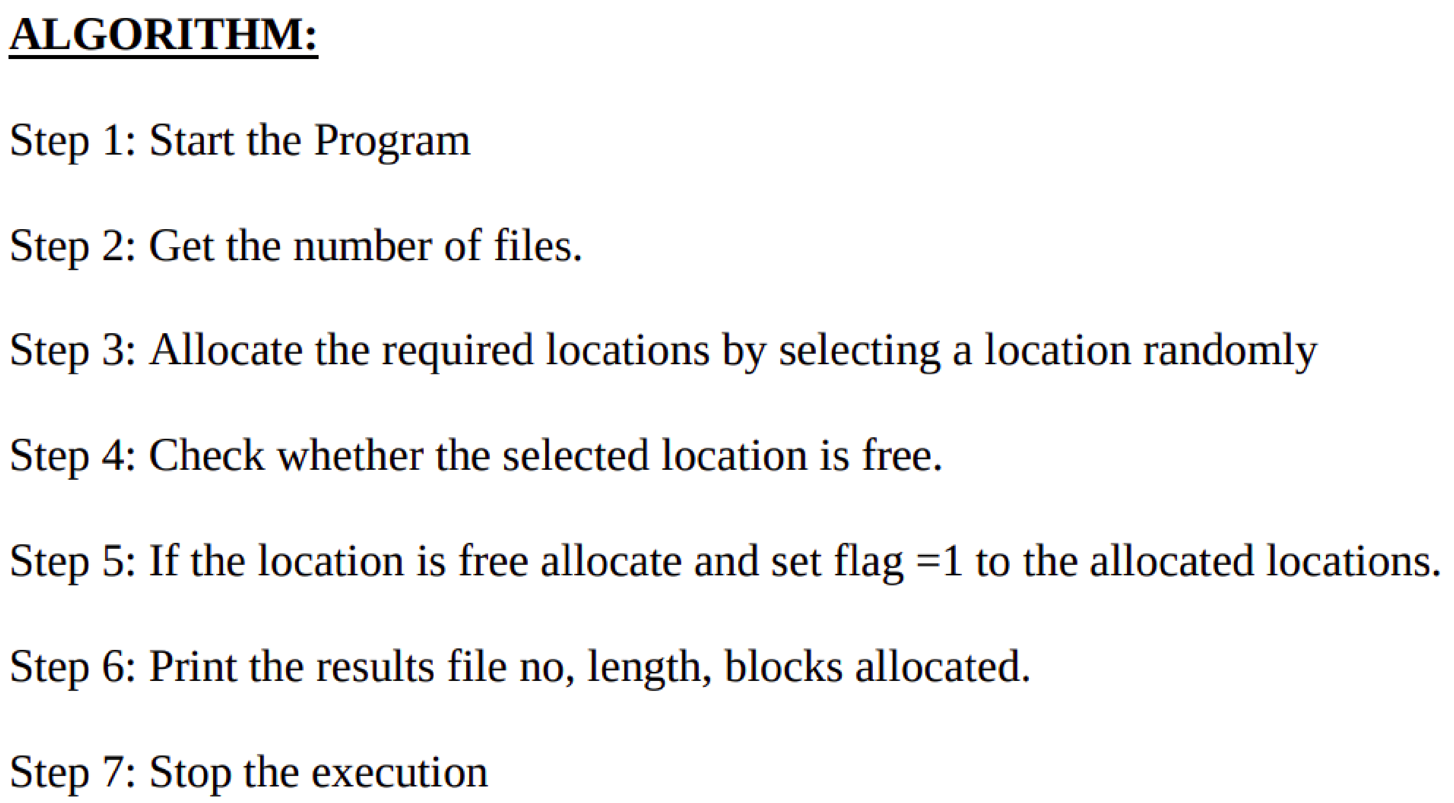
b). Check whether the required locations are free from the selected location.

c). Allocate and set flag=1 to the allocated locations.

Step 5: Print the results fileno, length , Blocks allocated.

Step 6: Stop the program.

Linked Allocation



ALGORITHM:

Step 1: Start the Program

Step 2: Get the number of files.

Step 3: Allocate the required locations by selecting a location randomly

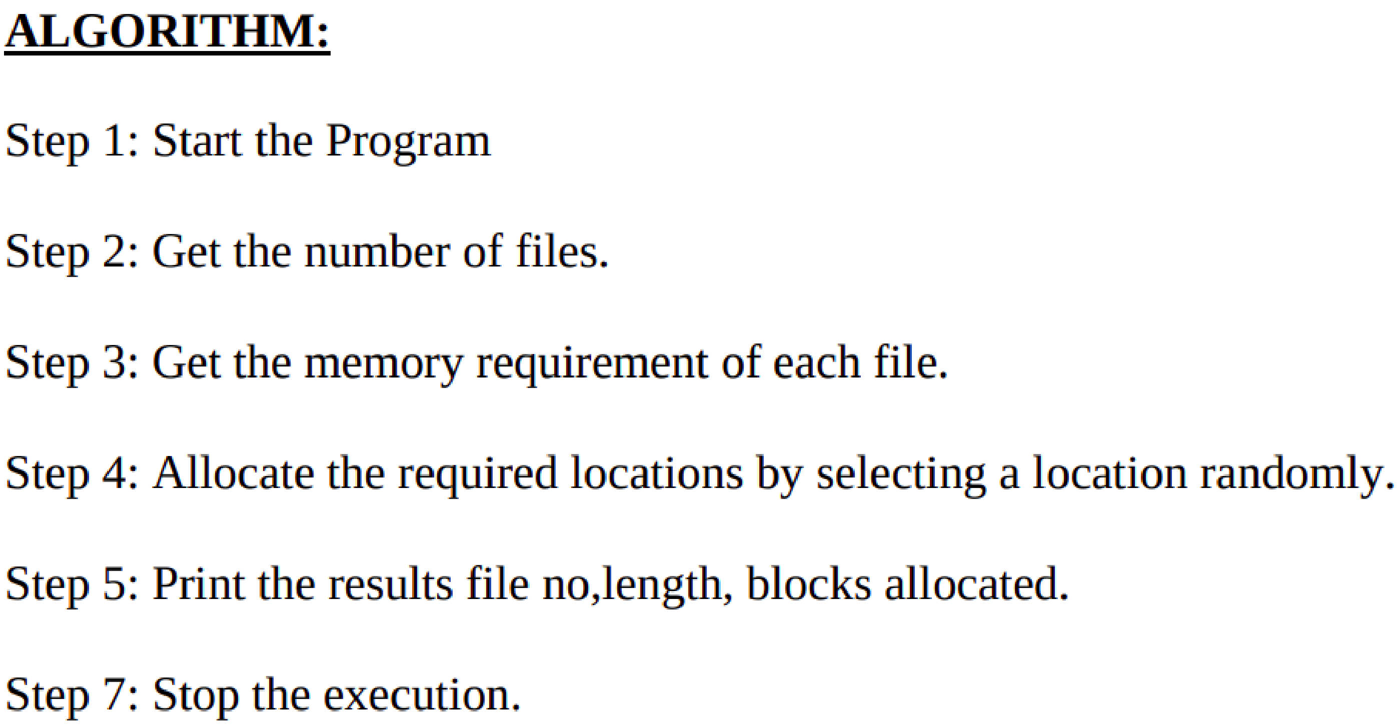
Step 4: Check whether the selected location is free.

Step 5: If the location is free allocate and set flag =1 to the allocated locations.

Step 6: Print the results file no, length, blocks allocated.

Step 7: Stop the execution

Indexed Allocation



ALGORITHM:

Step 1: Start the Program

Step 2: Get the number of files.

Step 3: Get the memory requirement of each file.

Step 4: Allocate the required locations by selecting a location randomly.

Step 5: Print the results file no,length, blocks allocated.

Step 7: Stop the execution.

FCFS Disk Scheduling Algorithm

1. Let Request array represents an array storing indexes of tracks that have been requested in ascending order of their time of arrival. ‘head’ is the position of disk head.
2. Let us one by one take the tracks in default order and calculate the absolute distance of the track from the head.
3. Increment the total seek count with this distance.
4. Currently serviced track position now becomes the new head position.
5. Go to step 2 until all tracks in the request array have not been serviced.

C-SCAN Disk Scheduling Algorithm

1. Let Request array represents an array storing indexes of tracks that have been requested in ascending order of their time of arrival. ‘head’ is the position of disk head.
2. The head services only in the right direction from 0 to the size of the disk.
3. While moving in the left direction do not service any of the tracks.
4. When we reach the beginning(left end) reverse the direction.
5. While moving in the right direction it services all tracks one by one.
6. While moving in the right direction calculate the absolute distance of the track from the head.
7. Increment the total seek count with this distance.
8. Currently serviced track position now becomes the new head position.
9. Go to step 6 until we reach the right end of the disk.
10. If we reach the right end of the disk, reverse the direction and go to step 3 until all tracks in the request array have not been serviced.

SCAN Disk Scheduling Algorithm

1. Let Request array represents an array storing indexes of tracks that have been requested in ascending order of their time of arrival. ‘head’ is the position of disk head.
2. Let direction represents whether the head is moving towards left or right.
3. In the direction in which head is moving service all tracks one by one.
4. Calculate the absolute distance of the track from the head.
5. Increment the total seek count with this distance.
6. Currently serviced track position now becomes the new head position.
7. Go to step 3 until we reach at one of the ends of the disk.
8. If we reach at the end of the disk reverse the direction and go to step 2 until all tracks in request array have not been serviced.