EXPERIMENT 10
MEMORY ALLOCATION SCHEMES

AIM
To implement the memory allocation schemes - First Fit, Best Fit & Worst fit memory allocation algorithms

ALGORITHM
STEP 0: START
STEP 1: Declare global variables with integer datatype psize[10], n, msize[10], m, i, j, flag, psize[10], msize[10], max & loc.
STEP 2: END

ALGORITHM FOR FIRSTFIT()
STEP 0: START
STEP 1: copy psize[] into psize[] using a for loop from i=0 till n.
STEP 2: copy MSIZE[] into msize[] using a for loop from i=0 till m.
STEP 3: Iterate through each process in the psize[] array.
STEP 4: Initialize flag = 0
STEP 5: Iterate through each memory block in the msize[] array.
STEP 6: check if the current memory block size is greater than or equal to

the current process size.
STEP 7: If suitable memory block is found then -
i) Print allocation message indicating the process size and memory block it has been processed to.
ii) Update the memory block by minus the process size.
iii) Set 'flag' to 1 to indicate allocation success and break out of inner loop.
STEP 8: If no suitable memory block is found (flag = 0), print message indicating the process cannot be allocated.
STEP 9: END.

ALGORITHM FOR WORSTFIT()
STEP 0: START
STEP 1: Copy the process size into psize[] array while in the for loop.
STEP 2: Copy the process msize[] into msize[] array while in the for loop.
STEP 3: Iterate through each process in the psize[] array.
STEP 4: Initialize 'max' to the size of first memory block and 'loc' to 0.
STEP 5: Iterate through each memory block in the msize[] array.

STEP 6: Find the memory block with maximum size ('maxi') & record its index ('loc')
STEP 7: Check if the maximum size found is greater than or equal to the current process size
STEP 8: If a suitable block is found then -
i) Print the allocation message indicating the process size and memory block its allocated to.
ii) Updated the memory block size by subtracting process size
STEP 9: If no suitable memory block is found, print message indicating the process cannot be allocated.
STEP 10: END.

ALHORITHM FOR BEST FIT()
STEP 0: START
STEP 1: copy the process PSIZE[] into the PSIZE[] array while in the for loop.
STEP 2: copy the memory size MSIZE[] into MSIZE[] array while in the for loop.
STEP 3: Iterate through each process in the PSIZE[] array.
STEP 4: Initialize 'loc' to -1.
STEP 5: Iterate through each memory block in the 'MSIZE[]' array.
STEP 6: Check if the current memory

block size is greater than or equal to the current size of process.
STEP 7: If a suitable memory block is found then -
i) If loc = -1, update loc to index of this memory block.
ii) If loc ≠ -1 and size of this memory block is smaller than the size of the memory block at loc, update loc to index of memory block.
STEP 8: If loc ≠ -1 after iterating through all memory blocks then -
i) Print allocation message indicating process size & memory block it is allocated to.
ii) Update the memory block by subtracting the process size.
STEP 9: If loc = -1, then print a message indicating the process cannot be allocated
STEP 10: END

ALGORITHM FOR MAIN()
STEP 0: START
STEP 1: Prompt user to enter the number of processes
STEP 2: Read no. of processes as n
STEP 3: Prompt user to enter the

sizes of each process and read
size of each process, store it in
psize[] array
STEP 4: Prompt user to enter number
of memory block and read the
no. of memory blocks as 'm'.
STEP 5: Prompt user to enter size of
each memory block and store them
in msize[] array.
STEP 6: Call the FIRSTFIT() function
to perform First Fit memory allocation
STEP 7: Call the BESTFIT() function
to perform Best Fit memory allocation
STEP 8: Call the WORSTFIT() function to
perform Worst Fit memory allocation
STEP 9: END.

RESULT
Experiment executed successfully
and output obtained.