20. Write a C program to implement the worst-fit algorithm and allocate the memory block to each process.

Test Case:

Memory partitions: 300 KB, 600 KB, 350 KB, 200 KB, 750 KB, and 125 KB (in order),

Show the outcome for the test case with the worst-fit algorithms to place processes of size 115 KB, 500 KB, 358 KB, 200 KB, and 375 KB (in order)

#include<stdio.h>

int main()

{

int partitions[] = {300, 600, 350, 200, 750, 125}; // memory partitions in KB

int num\_partitions = 6;

int processes[] = {115, 500, 358, 200, 375}; // processes in KB

int num\_processes = 5;

int allocated[num\_processes]; // array to store allocation status

int i, j, worst\_index;

// initialize allocation status to 0

for(i=0; i<num\_processes; i++)

allocated[i] = 0;

// allocate memory to each process using worst-fit algorithm

for(i=0; i<num\_processes; i++)

{

worst\_index = -1;

for(j=0; j<num\_partitions; j++)

{

if(partitions[j] >= processes[i])

{

if(worst\_index == -1 || partitions[j] > partitions[worst\_index])

worst\_index = j;

}

}

if(worst\_index != -1)

{

printf("Process %d of size %d KB allocated to partition %d of size %d KB\n", i+1, processes[i], worst\_index+1, partitions[worst\_index]);

partitions[worst\_index] -= processes[i];

allocated[i] = 1;

}

else

printf("Process %d of size %d KB cannot be allocated\n", i+1, processes[i]);

}

// display the remaining partition sizes

printf("\nRemaining partition sizes:\n");

for(i=0; i<num\_partitions; i++)

printf("%d. %d KB\n", i+1, partitions[i]);

return 0;

}

