Practical 13

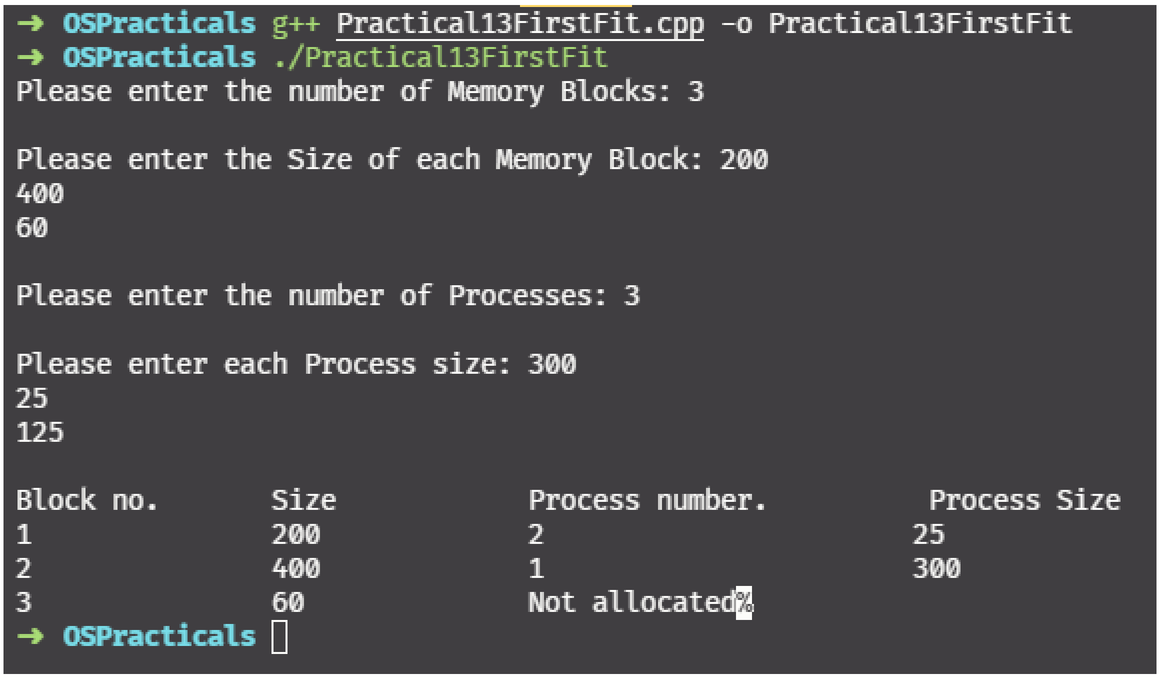
Write a program to implement first-fit, best-fit and worst-fit allocation strategies

CODE

(a) first fit

```
/* program to implement first-fit allocation strategies */

#include <iostream>
using namespace std;

int main()
{ // main function starts
    int MemoryBlock[10], Process[10], NumberOfBlock, NumberOfProcess,
flags[10],
        allocation[10], i, j;

    for (i = 0; i < 10; i++)
    { // updating initial allocation status
        flags[i] = 0;
        allocation[i] = -1;
    }

    cout << "Please enter the number of Memory Blocks: ";
    cin >> NumberOfBlock; // enter number of memory block

    cout << "\nPlease enter the Size of each Memory Block: ";
    for (i = 0; i < NumberOfBlock; i++)
    {
        cin >> MemoryBlock[i];
    } // enter size of each memory block

    cout << "\nPlease enter the number of Processes: ";
    cin >> NumberOfProcess; // enter number of processes

    cout << "\nPlease enter each Process size: ";
    for (i = 0; i < NumberOfProcess; i++)
    {
        cin >> Process[i];
    } // enter size of each process

    /* allocating according to first fit strategies */
    for (i = 0; i < NumberOfProcess;
         i++)
    { // comparing each process to each memory block
        for (j = 0; j < NumberOfBlock; j++)
        {
            if (flags[j] == 0 && MemoryBlock[j] >= Process[i])
            {
                /* if the mem block is not allocated and size of process is
less
                than mem block it will be allocated */

                allocation[j] = i; /* updating status of memory block to
                                      allocated and storing process number */
                flags[j] = 1;
                break;
            }
        }
    }
    /* displaying gannt chart table */
    cout << "\nBlock no.\tSize\t\tProcess number.\t\t Process Size";
    for (i = 0; i < NumberOfBlock; i++)
    {
        cout << "\n"
             << i + 1 << "\t\t" << MemoryBlock[i] << "\t\t";
        if (flags[i] == 1)
        {
            cout << allocation[i] + 1 << "\t\t\t" << Process[allocation[i]];
        }
        else
        {
            cout << "Not allocated";
        }
    }
    return 0;
}
```

OUTPUT



```
➜ OSPracticals g++ Practical13FirstFit.cpp -o Practical13FirstFit
➜ OSPracticals ./Practical13FirstFit
Please enter the number of Memory Blocks: 3

Please enter the Size of each Memory Block: 200
400
60

Please enter the number of Processes: 3

Please enter each Process size: 300
25
125

Block no.       Size            Process number.          Process Size
1               200             2                       25
2               400             1                       300
3               60              Not allocated%
➜ OSPracticals
```

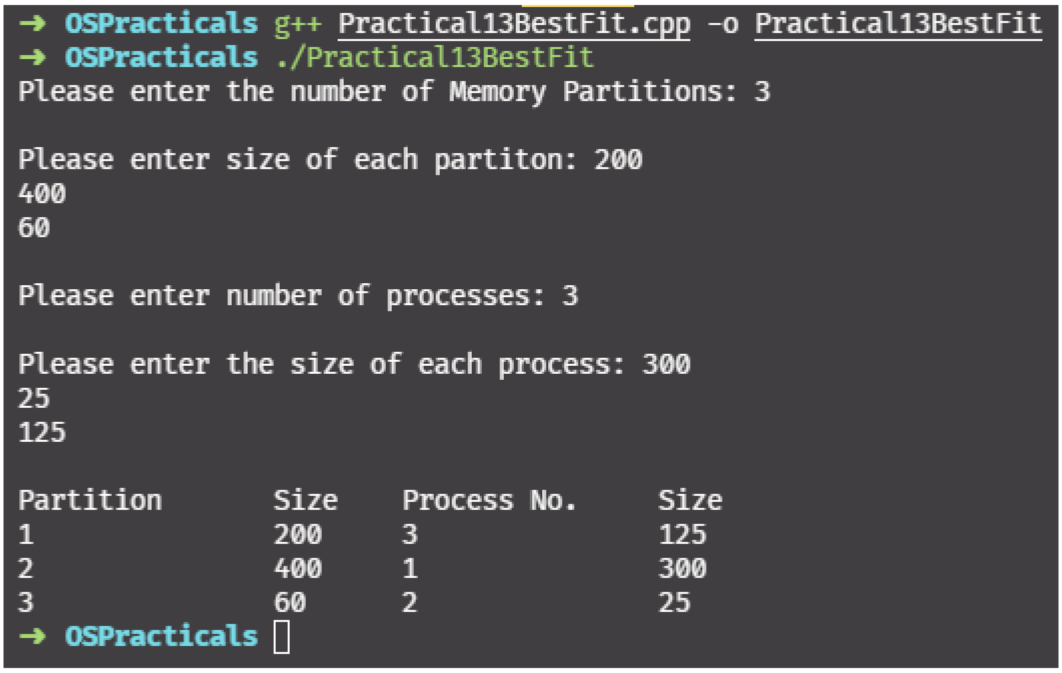
CODE

(b) best fit

```cpp
/* program to implement best-fit allocation strategies */

#include <iostream>  //input output
using namespace std; //standard namespace
int main()
{ //main function
    int MemoryBlock[10], Processes[10], numberOfMemoryBlocks, numberOfProc,
        flags[10], allocation[10];
    int i, j, smallest;
    //setting intial status of memory block to not allocated
    for (i = 0; i < 10; i++)
    {
        flags[i] = 0;
        allocation[i] = -1;
    }

    cout << "Please enter the number of Memory Partitions: ";
    cin >> numberOfMemoryBlocks; //enter number of mem block
    cout << "\nPlease enter size of each partiton: ";
    for (i = 0; i < numberOfMemoryBlocks; i++)
    {
        cin >> MemoryBlock[i];
    } //enter size of each memory block

    cout << "\nPlease enter number of processes: ";
    cin >> numberOfProc; //enter number of processess
    cout << "\nPlease enter the size of each process: ";
    for (i = 0; i < numberOfProc; i++)
    {
        cin >> Processes[i];
    } //enter size of each process

    // allocation as per best fit
    for (i = 0; i < numberOfProc; i++)
    {                   //comparing each process to each mem block
```

```
        smallest = -1; //initiating smallest memory block
        for (j = 0; j < numberOfMemoryBlocks; j++)
            if (flags[j] == 0 && MemoryBlock[j] >= Processes[i])
            {
                smallest = j;
                break;
            }
        for (j = 0; j < numberOfMemoryBlocks; j++)
        {
            if (flags[j] == 0 && MemoryBlock[j] >= Processes[i] &&
                MemoryBlock[j] < MemoryBlock[smallest])
                smallest = j;
        }
        if (smallest != -1)
        {
            allocation[smallest] = i;
            flags[smallest] = 1;
        }
    }

    /* displaying details */
    cout << "\nPartition\tSize\tProcess No.\tSize";
    for (i = 0; i < numberOfMemoryBlocks; i++)
    {
        cout << "\n"
             << i + 1 << "\t\t" << MemoryBlock[i] << "\t";
        if (flags[i] == 1)
            cout << allocation[i] + 1 << "\t\t" << Processes[allocation[i]];
        else
            cout << "Not allocated";
    }
    cout << endl;
    return 0;
}
```

OUTPUT



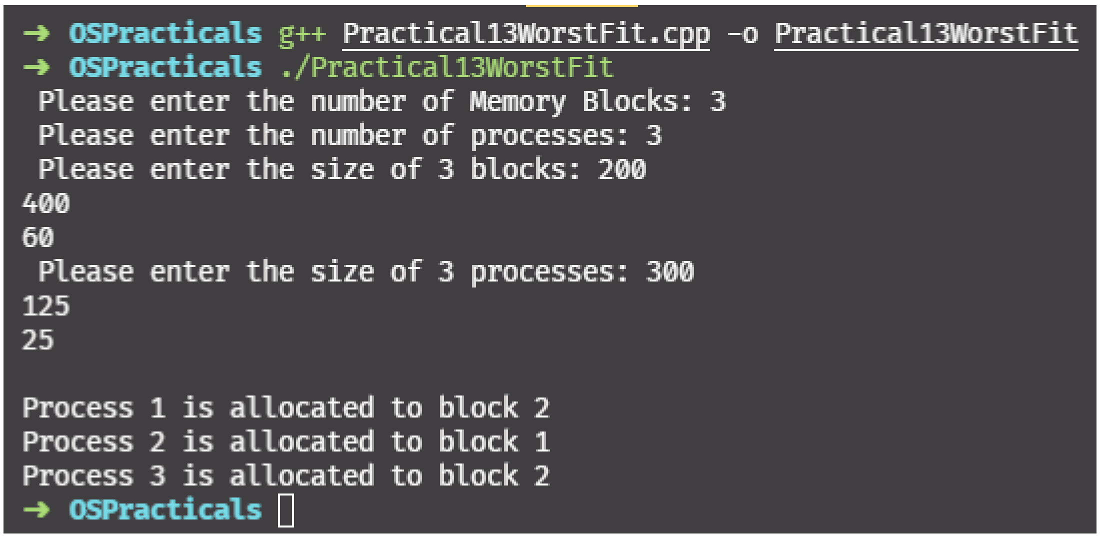
```
➜ OSPracticals g++ Practical13BestFit.cpp -o Practical13BestFit
➜ OSPracticals ./Practical13BestFit
Please enter the number of Memory Partitions: 3

Please enter size of each partiton: 200
400
60

Please enter number of processes: 3

Please enter the size of each process: 300
25
125

Partition       Size    Process No.     Size
1               200     3               125
2               400     1               300
3               60      2               25
➜ OSPracticals
```

CODE

(c) worst fit

```cpp
/* program to implement worst-fit allocation strategies */

#include <iostream>  //input output stream
using namespace std; // standard namespace

int main()
{ // main function
    int NumberOfBlock, NumberOfProcess, MemoryBlock[20], Processes[20];

    cout << " Please enter the number of Memory Blocks: ";
    cin >> NumberOfBlock; // enter number of blocks

    cout << " Please enter the number of processes: ";
    cin >> NumberOfProcess; // enter number of processes

    cout << " Please enter the size of " << NumberOfBlock << " blocks: ";
    for (int i = 0; i < NumberOfBlock; i++)
    {
        cin >> MemoryBlock[i];
    } // enter size of each mem block

    cout << " Please enter the size of " << NumberOfProcess << " processes: ";
    for (int i = 0; i < NumberOfProcess; i++)
    {
        cin >> Processes[i];
    } // enter size of each processes

    // performing worst fit allocation strategies
    for (int i = 0; i < NumberOfProcess; i++)
    {
        /* comparing each process with each memory block */
        int max = MemoryBlock[0];
        int pos = 0;
        for (int j = 0; j < NumberOfBlock; j++)
            if (max < MemoryBlock[j])
            {
                max = MemoryBlock[j];
                pos = j;
            }
        /* displaying details */
        if (max >= Processes[i])
        {
            cout << "\nProcess " << i + 1 << " is allocated to block "
                 << pos + 1;
            MemoryBlock[pos] = MemoryBlock[pos] - Processes[i];
        }
        else
        {
            cout << "\nProcess " << i + 1 << " can't be allocated!";
        }
    }
    cout << endl;
    return 0;
}
```

OUTPUT



```
➜ OSPracticals g++ Practical13WorstFit.cpp -o Practical13WorstFit
➜ OSPracticals ./Practical13WorstFit
 Please enter the number of Memory Blocks: 3
 Please enter the number of processes: 3
 Please enter the size of 3 blocks: 200
400
60
 Please enter the size of 3 processes: 300
125
25

Process 1 is allocated to block 2
Process 2 is allocated to block 1
Process 3 is allocated to block 2
➜ OSPracticals
```