\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Operating System Lab

CEN-493

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Program - 12

Code :-

#include <iostream>

#include <vector>

using namespace std;

typedef long long ll;

struct memoryBlocks

{

    bool isAllocated;

    int blockSize;

    int processSize;

    int internalFrag;

    string processName;

};

void printLines()

{

    for (int i = 0; i < 110; i++)

    {

        cout << "\_";

    }

    cout << "\n";

}

void Display(vector<memoryBlocks> &memBlocks, int noOfBlocks, int internalFrag, int externalFrag, vector<pair<int, string>> &leftProcess)

{

    cout << "Worst Fit Memory Allocation Table \n";

    cout << "---------------------------------------------------------------------------------\n";

    cout << "| Block No\t"

         << "Size Of Block\t"

         << "Proces Allocated\t"

         << "Internal Fragmentation  |\n";

    cout << "---------------------------------------------------------------------------------\n";

    for (int bindx = 0; bindx < noOfBlocks; bindx++)

    {

        if (memBlocks[bindx].isAllocated == false)

            cout << "|  " << bindx + 1 << "\t\t\t" << memBlocks[bindx].blockSize << "\t\t"

                 << "  ---  "

                 << "\t\t\t"

                 << "--"

                 << "\t\t|\n";

        else

            cout << "|  " << bindx + 1 << "\t\t\t" << memBlocks[bindx].blockSize << "\t\t"

                 << memBlocks[bindx].processSize << "[" << memBlocks[bindx].processName << "]"

                 << "\t\t\t" << memBlocks[bindx].internalFrag << "\t\t|\n";

    }

    cout << "---------------------------------------------------------------------------------\n";

    cout << "\n";

    printLines();

    printLines();

    if (!leftProcess.empty())

    {

        cout << "Process Whom Memory Is Not Allocated : \n";

        for (int lindx = 0; lindx < leftProcess.size(); lindx++)

        {

            cout << leftProcess[lindx].second << " " << leftProcess[lindx].first << "\n";

        }

    }

    printLines();

    cout << "\n\n";

    printLines();

    cout << "Total Internal Fragmentation = " << internalFrag << "\n";

    cout << "Total External Fragmentation = " << externalFrag << "\n";

    printLines();

}

void Worst\_Fit(vector<memoryBlocks> &memBlocks, int noOfBlocks, vector<pair<int, string>> &processSizes, int noOfProcess)

{

    vector<pair<int, string>> leftProcess;

    for (int pindx = 0; pindx < noOfProcess; pindx++)

    {

        bool isProcessMemAllocated = false;

        int emptyBlock = 0, largestBlockSize = 0;

        for (int bindx = 0; bindx < noOfBlocks; bindx++)

        {

            if (memBlocks[bindx].isAllocated == true || memBlocks[bindx].blockSize < processSizes[pindx].first)

                continue;

            isProcessMemAllocated = true;

            if (largestBlockSize < memBlocks[bindx].blockSize)

            {

                emptyBlock = bindx;

                largestBlockSize = memBlocks[bindx].blockSize;

            }

        }

        if (isProcessMemAllocated == false)

        {

            leftProcess.push\_back(processSizes[pindx]);

        }

        else

        {

            memBlocks[emptyBlock].isAllocated = true;

            memBlocks[emptyBlock].processName = processSizes[pindx].second;

            memBlocks[emptyBlock].processSize = processSizes[pindx].first;

            memBlocks[emptyBlock].internalFrag = memBlocks[emptyBlock].blockSize - processSizes[pindx].first;

        }

    }

    int externalFrag = 0, internalFrag = 0;

    if (leftProcess.empty() == false)

    {

        for (int bindx = 0; bindx < noOfBlocks; bindx++)

        {

            if (memBlocks[bindx].isAllocated == true)

                continue;

            externalFrag += memBlocks[bindx].blockSize;

        }

        int leftProcessSize = 0;

        bool isExternFrag = 0;

        for (int iter = 0; iter < leftProcess.size(); iter++)

        {

            if (leftProcess[iter].first < externalFrag)

            {

                isExternFrag = 1;

                break;

            }

        }

        if (isExternFrag == 0)

        {

            externalFrag = 0;

        }

    }

    for (int bindx = 0; bindx < noOfBlocks; bindx++)

    {

        internalFrag += memBlocks[bindx].internalFrag;

    }

    Display(memBlocks, noOfBlocks, internalFrag, externalFrag, leftProcess);

}

int main()

{

    system("cls");

    printLines();

    cout << "Vicky Gupta 20BCS070\n";

    cout << "Worst Fit Memory Allocation Algorithm\n";

    printLines();

    printLines();

    int noOfBlocks;

    cout << "Enter The No Of Blocks Of Memory : ";

    cin >> noOfBlocks;

    printLines();

    int noOfProcess;

    cout << "Enter The No Of Process : ";

    cin >> noOfProcess;

    printLines();

    vector<memoryBlocks> memBlocks(noOfBlocks);

    cout << "Enter The Sizes Of Blocks : ";

    for (int i = 0; i < noOfBlocks; i++)

    {

        cin >> memBlocks[i].blockSize;

        memBlocks[i].isAllocated = false;

        memBlocks[i].processSize = 0;

        memBlocks[i].processName = "";

        memBlocks[i].internalFrag = 0;

    }

    printLines();

    vector<pair<int, string>> processSizes(noOfProcess);

    cout << "Enter The Sizes Of Process : ";

    for (int i = 0; i < noOfProcess; i++)

    {

        cin >> processSizes[i].first;

        processSizes[i].second = "P";

        processSizes[i].second += to\_string(i + 1);

    }

    printLines();

    cout << "Memory Blocks...\n";

    cout << "| ";

    for (int i = 0; i < noOfBlocks; i++)

    {

        cout << memBlocks[i].blockSize << " | ";

    }

    cout << "\n";

    printLines();

    cout << "Process Blocks...\n";

    cout << "| ";

    for (int i = 0; i < noOfProcess; i++)

    {

        cout << processSizes[i].first << " [" << processSizes[i].second << "]  | ";

    }

    cout << "\n\n";

    printLines();

    printLines();

    Worst\_Fit(memBlocks, noOfBlocks, processSizes, noOfProcess);

    return 0;

}

Output :-

Graphical user interface

Description automatically generated with low confidence