FORM NO. SSGMCE/FRM/32 B

	SHRI SANT GAJANAN MAHARAJ COLLEGE ENGG. SHEGAON		LABORATORY MANUAL
	PRACTICAL EXPERIMENT INSTRUCTION SHEET		
	EXPERIMENT TITLE: **First Fit Contiguous Memory Allocation**		
EXPERIMENT NO.: SSGMCE/WI/ASH/01/1A6/01		ISSUE NO.:	**ISSUE DATE:**26.08.23
REV. DATE: 26.08.23	REV. NO.: 00	DEPTT. : INFORMATION TECHNOLOGY.	
LABORATORY: OPERATING SYSTEMS		SEMESTER : IV	PAGE NO. 01 OF 05

Date:

FIRST FIT CONTIGUOUS MEMORY ALLOCATION

01. AIM :

Write a program to implement First Fit contiguous memory allocation algorithms.

02. FACILITIES :

Hardware: A Linux-based system (e.g., Ubuntu, Fedora, or any other distribution).

Software: C programming language, GCC compiler for compiling and running the program.

IDE/Editor: Any text editor or IDE that supports C programming (e.g., Visual Studio Code, Vim, Sublime Text, or Eclipse).

03. SCOPE :

This experiment aims to simulate memory allocation in an operating system using the **First Fit** algorithm. The scope includes:

- Allocating memory blocks to processes using the First Fit strategy.
- Understanding memory fragmentation and allocation in contiguous memory allocation schemes.

04 THEORY:

Memory Management in Operating System:

The term memory can be defined as a collection of data in a specific format. It is used to store instructions and process data. The memory comprises a large array or group of words or bytes, each with its own location.

The primary purpose of a computer system is to execute programs. These programs, along with the information they access, should be in the main memory during execution. The CPU fetches instructions from memory according to the value of the program counter.

To achieve a degree of multiprogramming and proper utilization of memory, memory management is important. Many memory management methods exist, reflecting various approaches, and the effectiveness of each algorithm depends on the situation.

Key Functions of Memory Management:

- **Memory Allocation:** Allocating memory to different processes based on their requirements.
- **Memory Deallocation:** Releasing memory that is no longer in use.
- **Memory Protection:** Ensuring that processes do not access or alter memory spaces assigned to other processes.
- **Memory Sharing:** Allowing processes to share memory if necessary (for example, shared libraries).
- **Swapping:** Moving processes between main memory and disk storage to free up memory.

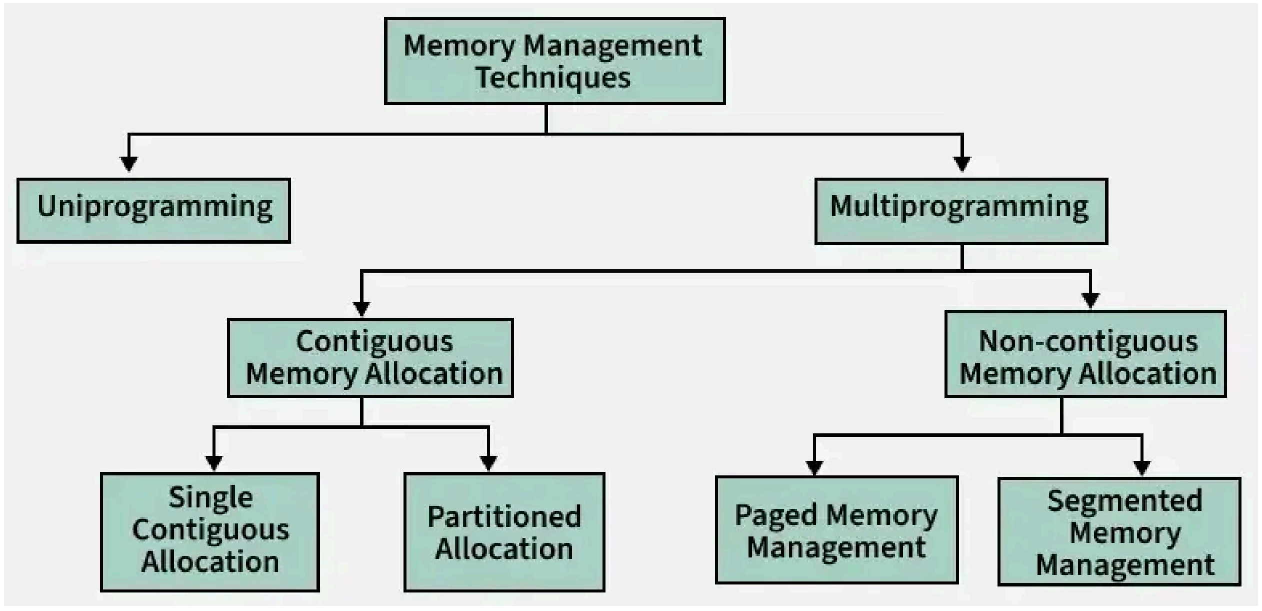
Why is Memory Management Required?

1. Allocate and de-allocate memory before and after process execution.
2. To keep track of used memory space by processes.
3. To minimize fragmentation issues.
4. To proper utilization of main memory.
5. To maintain data integrity while executing the process.

Memory Management Techniques:

Memory management techniques are methods used by an operating system to efficiently allocate, utilize, and manage memory resources for processes. These techniques ensure smooth execution of programs and optimal use of system memory

Different Memory Management techniques are:


Memory Management Techniques
Uniprogramming
Multiprogramming
Contiguous Memory Allocation
Non-contiguous Memory Allocation
Single Contiguous Allocation
Partitioned Allocation
Paged Memory Management
Segmented Memory Management

First-Fit Memory Allocation:

The **First-Fit** memory allocation is a dynamic memory allocation scheme used in contiguous memory allocation. The basic idea of the First-Fit algorithm is to allocate the first available memory block that is large enough to accommodate a process.

In First-Fit, the operating system searches through the list of free blocks of memory, starting from the beginning of the list, until it finds a block that is large enough to accommodate the memory request from the process. Once a suitable block is found, the operating system splits the block into two parts: the portion that will be allocated to the process, and the remaining free block.

How First-Fit Memory Allocation Works:

1. The system maintains a list of memory blocks available for allocation.
2. When a process requests memory, the system scans the list of available memory blocks.
3. The first block that is large enough to hold the process is selected and allocated.
4. The remaining portion of the block (if any) is split into a new smaller block and is added back to the list of available memory.

Advantages of First-Fit Allocation:

1. Simple and efficient search algorithm
2. Minimizes memory fragmentation
3. Fast allocation of memory

Disadvantages of First-Fit Allocation:

1. Poor performance in highly fragmented memory
2. May lead to poor memory utilization
3. May allocate larger blocks of memory than required.

Example:

Imagine a system with memory blocks of sizes [100KB, 300KB, 500KB, 200KB]. If a process requests 250KB:

1. The system scans the available blocks and finds that the first block that is large enough is 300KB.
2. The 250KB is allocated to this block, and the remaining 50KB is left free.
3. The new free block of 50KB is added to the list of available blocks.

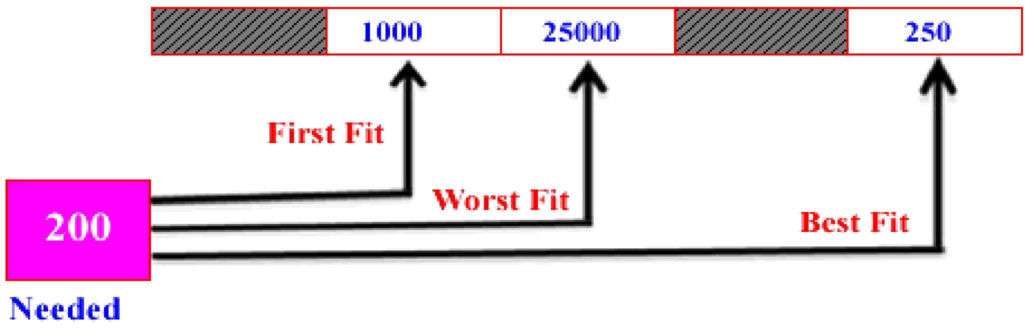
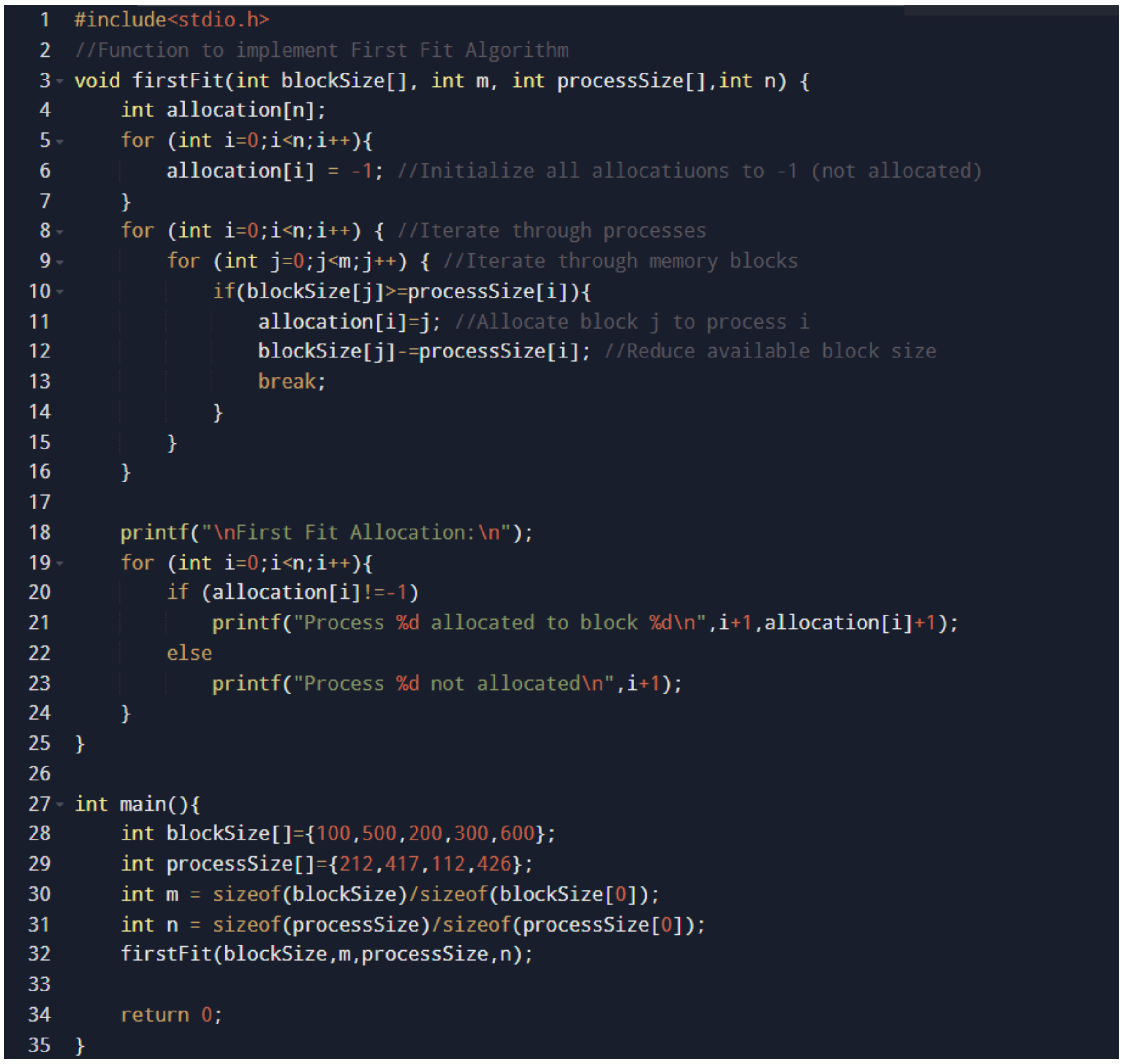


Fig: Showing First Fit, Worst Fit and Best Fit Memory Allocations

05 PROGRAM:



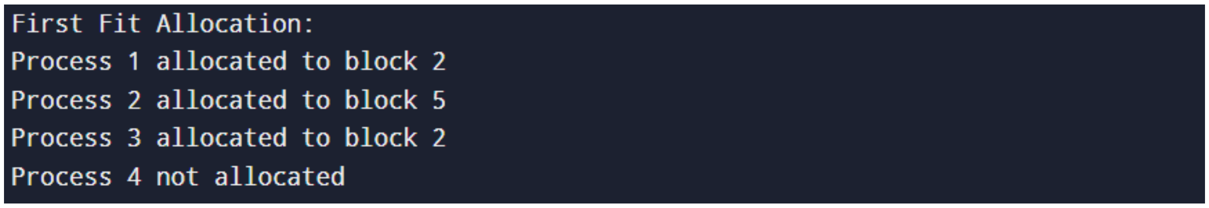
```c
#include<stdio.h>
//Function to implement First Fit Algorithm
void firstFit(int blockSize[], int m, int processSize[],int n) {
    int allocation[n];
    for (int i=0;i<n;i++){
        allocation[i] = -1; //Initialize all allocatiuons to -1 (not allocated)
    }
    for (int i=0;i<n;i++) { //Iterate through processes
        for (int j=0;j<m;j++) { //Iterate through memory blocks
            if(blockSize[j]>=processSize[i]){
                allocation[i]=j; //Allocate block j to process i
                blockSize[j]-=processSize[i]; //Reduce available block size
                break;
            }
        }
    }

    printf("\nFirst Fit Allocation:\n");
    for (int i=0;i<n;i++){
        if (allocation[i]!=-1)
            printf("Process %d allocated to block %d\n",i+1,allocation[i]+1);
        else
            printf("Process %d not allocated\n",i+1);
    }
}

int main(){
    int blockSize[]={100,500,200,300,600};
    int processSize[]={212,417,112,426};
    int m = sizeof(blockSize)/sizeof(blockSize[0]);
    int n = sizeof(processSize)/sizeof(processSize[0]);
    firstFit(blockSize,m,processSize,n);

    return 0;
}
```

06 OUTPUT



```
First Fit Allocation:
Process 1 allocated to block 2
Process 2 allocated to block 5
Process 3 allocated to block 2
Process 4 not allocated
```

08 CONCLUSION:

The **First Fit** algorithm is simple and quick, suitable for scenarios where memory allocation is straightforward. While it is effective in small systems, it may result in fragmentation over time. This experiment highlights the trade-offs in memory management strategies and demonstrates the allocation process in a clear and efficient manner.

09 VIVA QUESTIONS:

PREPARED BY **H. P. Amle**	**APPROVED BY** **HOD**