

**Name:** A. HARSHA MUKUNDHA

**Reg-No:** 192324070

23. Construct a C program to implement the first fit algorithm of memory management.

### **Aim**

To implement the First Fit memory allocation algorithm to allocate memory to processes based on their requirements.

### **Algorithm**

1. **Input:**
  - Number of memory blocks and their sizes.
  - Number of processes and their memory requirements.
2. For each process, traverse the memory blocks list sequentially.
3. Assign the process to the first memory block that is large enough to satisfy its requirement.
4. If a process cannot find a suitable block, it remains unallocated.
5. Display the allocation result for each process.

### **Procedure**

1. Initialize arrays for memory block sizes, process sizes, and allocation status.
2. For each process:
  - Check memory blocks sequentially.
  - If a block can satisfy the process size and is free, allocate the process to the block.
3. Print the allocation results.

### **Code:**

```
#include <stdio.h>
```

```
int main() {  
  
    int nBlocks, nProcesses;  
  
    printf("Enter the number of memory blocks: ");  
  
    scanf("%d", &nBlocks);  
  
    int blockSize[nBlocks], blockAllocated[nBlocks];
```

```
printf("Enter the sizes of the memory blocks: ");

for (int i = 0; i < nBlocks; i++) {

    scanf("%d", &blockSize[i]);

    blockAllocated[i] = 0;

}


printf("Enter the number of processes: ");

scanf("%d", &nProcesses);

int processSize[nProcesses], processAllocated[nProcesses];

printf("Enter the sizes of the processes: ");

for (int i = 0; i < nProcesses; i++) {

    scanf("%d", &processSize[i]);

    processAllocated[i] = -1;

}


for (int i = 0; i < nProcesses; i++) {

    for (int j = 0; j < nBlocks; j++) {

        if (blockSize[j] >= processSize[i] && blockAllocated[j] == 0) {

            blockAllocated[j] = 1;

            processAllocated[i] = j;

            break;

        }

    }

}
```

```

    }

    printf("\nProcess\tSize\tBlock Allocated\n");

    for (int i = 0; i < nProcesses; i++) {

        printf("%d\t%d\t", i + 1, processSize[i]);

        if (processAllocated[i] != -1)

            printf("%d\n", processAllocated[i] + 1);

        else

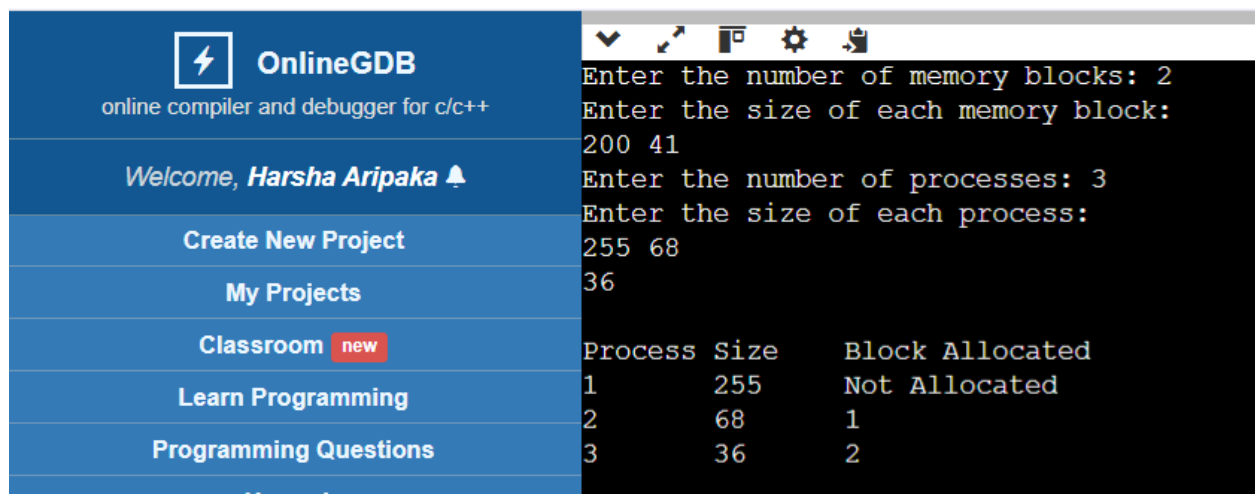
            printf("Not Allocated\n");

    }

    return 0;
}

```

### Output:



The screenshot shows the OnlineGDB interface. On the left is a sidebar with the OnlineGDB logo and navigation links: 'Welcome, Harsha Aripaka', 'Create New Project', 'My Projects', 'Classroom' (marked as new), 'Learn Programming', and 'Programming Questions'. The main area on the right contains a terminal window with the following output:

```

Enter the number of memory blocks: 2
Enter the size of each memory block:
200 41
Enter the number of processes: 3
Enter the size of each process:
255 68
36

```

Process	Size	Block Allocated
1	255	Not Allocated
2	68	1
3	36	2