B. Siva Shirish-192324016

22.**Construct a C program to implement the best fit algorithm of memory management.**

Aim:

To implement the Best Fit memory allocation algorithm in C, which allocates memory blocks to processes such that the block with the smallest size sufficient for the process is selected.

Algorithm:

1. **Input:** Sizes of memory blocks and processes.
2. **Sort:** Go through each process and find the smallest memory block that fits.
3. **Allocation:**
    - If a suitable block is found, allocate it to the process and update the memory block size.
    - If no suitable block is found, mark the process as unallocated.
4. **Output:** Display allocation details for each process.

Procedure:

1. Define the sizes of memory blocks and processes.
2. Iterate over each process.
3. For each process, check all memory blocks to find the smallest block that fits.
4. Allocate the block, and update its size or mark the process as unallocated.
5. Display the allocation results.

Code:

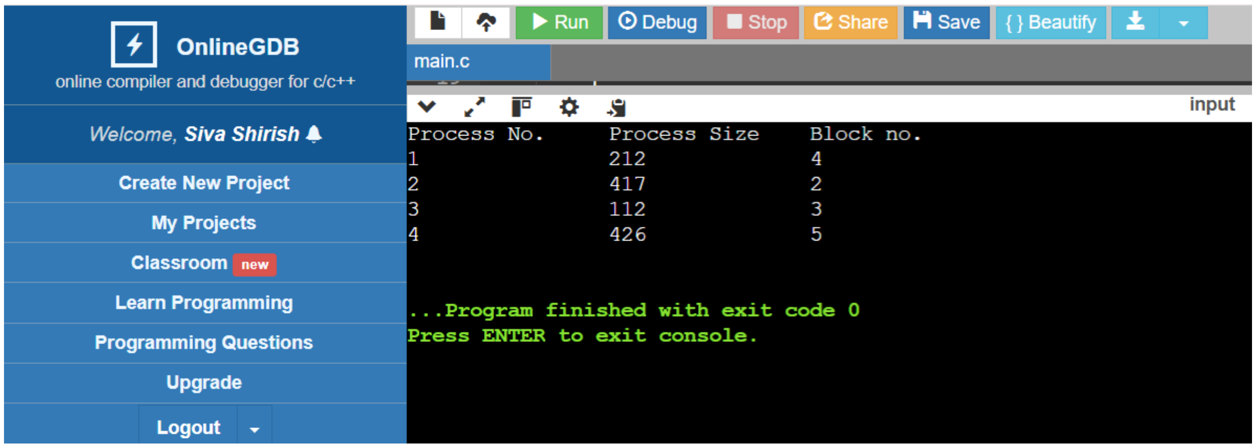
```
#include <stdio.h>



#define MAX 100



void bestFit(int blockSize[], int m, int processSize[], int n) {

    int allocation[n];

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

        allocation[i] = -1;
```

```
for (int i = 0; i < n; i++) {
    int bestIdx = -1;
    for (int j = 0; j < m; j++) {
        if (blockSize[j] >= processSize[i]) {
            if (bestIdx == -1)
                bestIdx = j;
            else if (blockSize[bestIdx] > blockSize[j])
                bestIdx = j;
        }
    }
    if (bestIdx != -1) {
        allocation[i] = bestIdx;
        blockSize[bestIdx] -= processSize[i];
    }
}

printf("Process No.\tProcess Size\tBlock no.\n");
for (int i = 0; i < n; i++) {
    printf("%d\t\t%d\t\t", i + 1, processSize[i]);
    if (allocation[i] != -1)
        printf("%d\n", allocation[i] + 1);
        else
            printf("Not Allocated\n");
    }
}


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]);
    bestFit(blockSize, m, processSize, n);
    return 0;
}
```

Result:

The program successfully implements the Best Fit algorithm. It allocates memory blocks to processes optimally, minimizing wastage by selecting the smallest block that fits each process.

Output:



```
Process No.     Process Size    Block no.
1               212             4
2               417             2
3               112             3
4               426             5


...Program finished with exit code 0
Press ENTER to exit console.
```