main.c

Run

```c
#include <stdio.h>

#define NUM_PARTITIONS 6
#define NUM_PROCESSES 5

int main() {
    int partitions[NUM_PARTITIONS] = {300, 600, 350, 200, 750, 125};
    int processes[NUM_PROCESSES] = {115, 500, 358, 200, 375};
    int allocation[NUM_PROCESSES];
    int i, j;

    // Initialize allocation array to -1 (unallocated)
    for (i = 0; i < NUM_PROCESSES; i++) {
        allocation[i] = -1;
    }

    // Allocate memory to each process
    for (i = 0; i < NUM_PROCESSES; i++) {
        int best_index = -1;
        for (j = 0; j < NUM_PARTITIONS; j++) {
            if (partitions[j] >= processes[i]) {
                if (best_index == -1 || partitions[j] <
                    partitions[best_index]) {
                    best_index = j;
                }
            }
        }
        if (best_index != -1) {
```

Output

Clear

```
/tmp/gUw86pWNpZ.o
Process Size    Partition
1       115     6
2       500     2
3       358     5
4       200     4
5       375     5
```

main.c

Run

```
    for (j = 0; j < NUM_PARTITIONS; j++) {
        if (partitions[j] >= processes[i]) {
            if (best_index == -1 || partitions[j] <
                partitions[best_index]) {
                best_index = j;
            }
        }
    }
    if (best_index != -1) {
        allocation[i] = best_index;
        partitions[best_index] -= processes[i];
    }
}

// Print the allocation
printf("Process\tSize\tPartition \n");
for (i = 0; i < NUM_PROCESSES; i++) {
    printf("%d\t  %d\t", i+1, processes[i]);
    if (allocation[i] != -1) {
        printf("%d\n", allocation[i]+1);
    } else {
        printf("Not allocated\n");
    }
}

return 0;
}
```

Output

Clear

```
/tmp/gUw86pWNpZ.o
Process Size    Partition
1       115     6
2       500     2
3       358     5
4       200     4
5       375     5
```