Name : Sujal Santosh Porte
PRN : 122B1B227

Assignment No-6

Code :

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

// Function to display the memory allocation
void displayAllocation(int allocation[], int processSize[], int n) {
    printf("\nProcess No.\tProcess Size\tBlock No.\n");
    for (int i = 0; i < n; i++) {
        printf(" %i\t\t\t", i + 1);
        printf("%i\t\t\t\t", processSize[i]);
        if (allocation[i] != -1)
            printf("%i", allocation[i] + 1);
        else
            printf("Not Allocated");
        printf("\n");
    }
}

// First Fit memory allocation
void firstFit(int blockSize[], int m, int processSize[], int n) {
    int allocation[n];
    for (int i = 0; i < n; i++)
        allocation[i] = -1;

    int tempBlocks[m];
    for (int i = 0; i < m; i++)
        tempBlocks[i] = blockSize[i];

    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            if (tempBlocks[j] >= processSize[i]) {
                allocation[i] = j;
                tempBlocks[j] -= processSize[i];
                break;
            }
        }
    }
    displayAllocation(allocation, processSize, n);
```

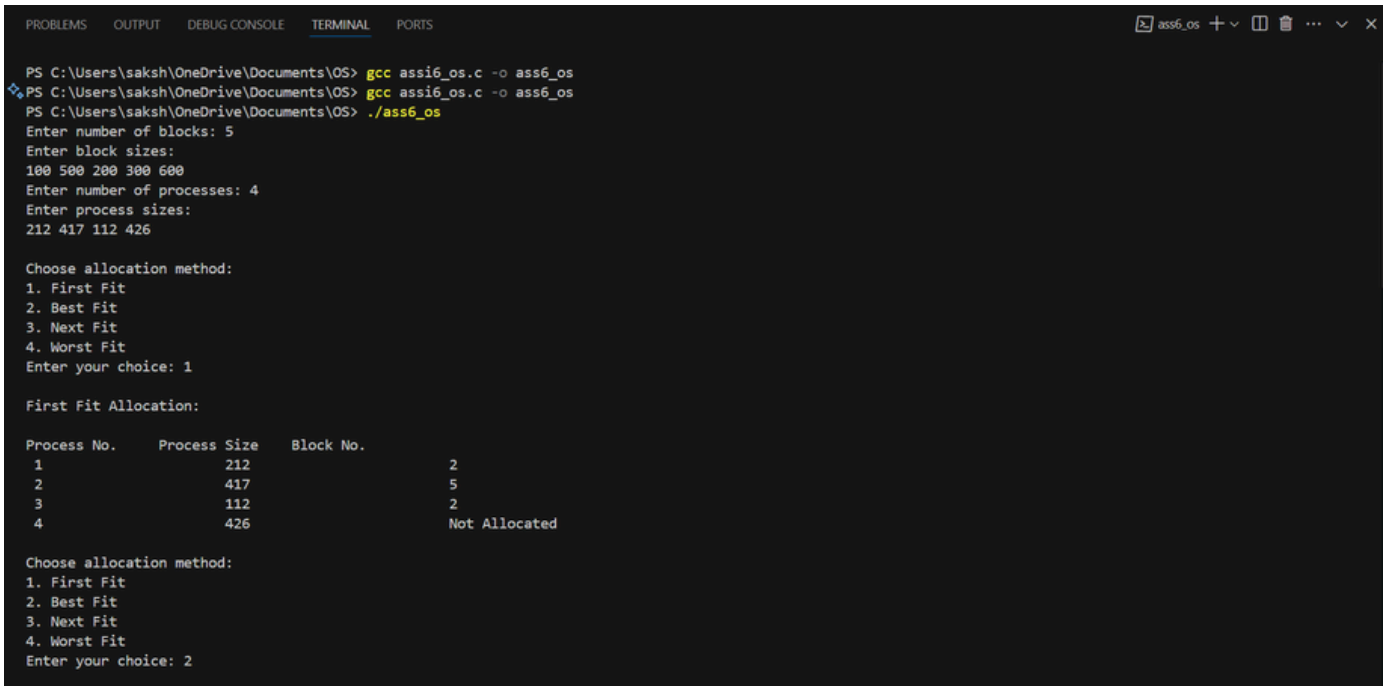
```
}

// Best Fit memory allocation
void bestFit(int blockSize[], int m, int processSize[], int n) {
    int allocation[n];
    for (int i = 0; i < n; i++)
        allocation[i] = -1;

    int tempBlocks[m];
    for (int i = 0; i < m; i++)
        tempBlocks[i] = blockSize[i];

    for (int i = 0; i < n; i++) {
        int bestIdx = -1, minDiff = INT_MAX;
        for (int j = 0; j < m; j++) {
            if (tempBlocks[j] >= processSize[i] && tempBlocks[j] - processSize[i] < minDiff &&
allocation[i] == -1) {
                minDiff = tempBlocks[j] - processSize[i];
                bestIdx = j;
            }
        }
        if (bestIdx != -1) {
            allocation[i] = bestIdx;
            tempBlocks[bestIdx] -= processSize[i];
        }
    }
    displayAllocation(allocation, processSize, n);
}

// Next Fit memory allocation
void nextFit(int blockSize[], int m, int processSize[], int n) {
    int allocation[n];
    for (int i = 0; i < n; i++)
        allocation[i] = -1;

    int tempBlocks[m];
    for (int i = 0; i < m; i++)
        tempBlocks[i] = blockSize[i];

    int lastAllocated = 0;
    for (int i = 0; i < n; i++) {
        int j = lastAllocated;
        do {
            if (tempBlocks[j] >= processSize[i] && allocation[i] == -1) {
```

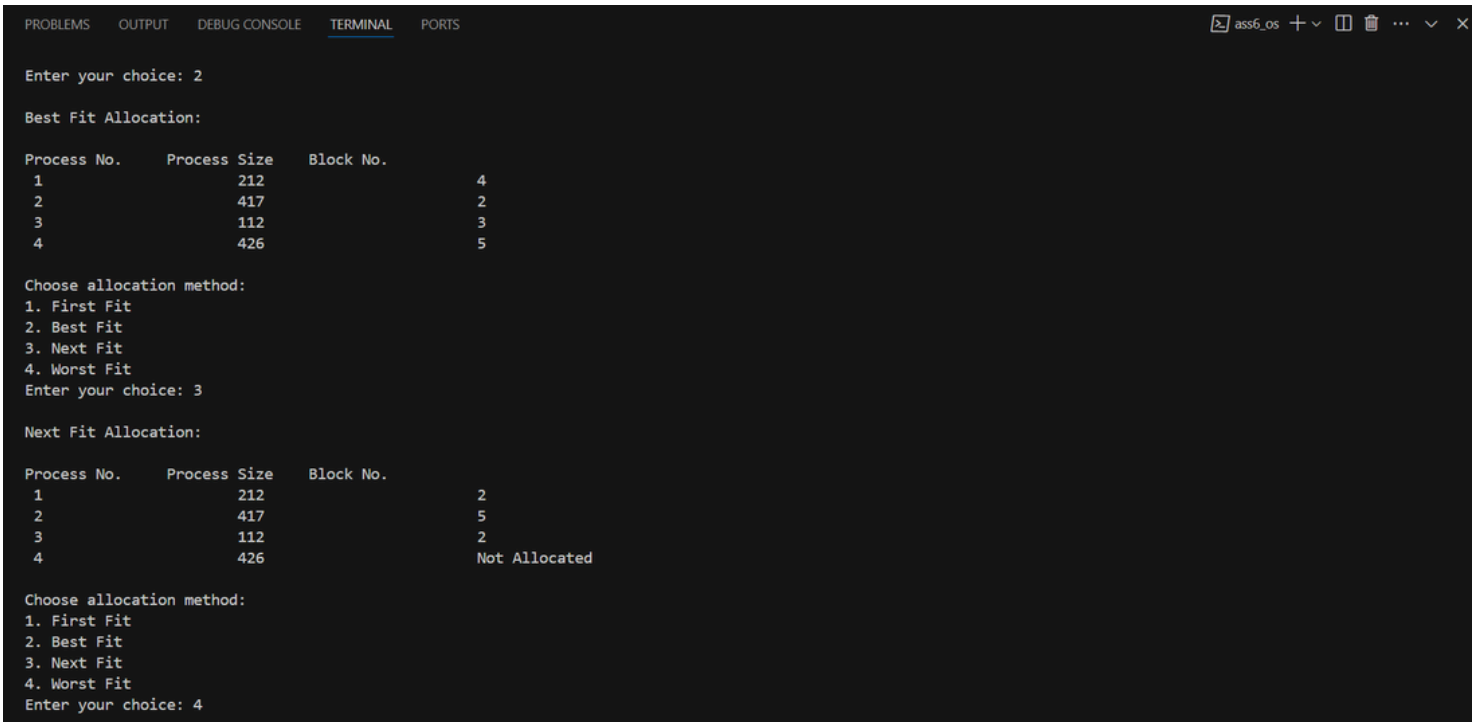
```
                allocation[i] = j;
                tempBlocks[j] -= processSize[i];
                lastAllocated = (j + 1) % m;
                break;
            }
            j = (j + 1) % m;
        } while (j != lastAllocated);
    }
    displayAllocation(allocation, processSize, n);
}

// Worst Fit memory allocation
void worstFit(int blockSize[], int m, int processSize[], int n) {
    int allocation[n];
    for (int i = 0; i < n; i++)
        allocation[i] = -1;

    int tempBlocks[m];
    for (int i = 0; i < m; i++)
        tempBlocks[i] = blockSize[i];

    for (int i = 0; i < n; i++) {
        int worstIdx = -1, maxDiff = INT_MIN;
        for (int j = 0; j < m; j++) {
            if (tempBlocks[j] >= processSize[i] && tempBlocks[j] - processSize[i] > maxDiff &&
allocation[i] == -1) {
                maxDiff = tempBlocks[j] - processSize[i];
                worstIdx = j;
            }
        }
        if (worstIdx != -1) {
            allocation[i] = worstIdx;
            tempBlocks[worstIdx] -= processSize[i];
        }
    }
    displayAllocation(allocation, processSize, n);
}

int main() {
    int m, n, choice;
    int *blockSize, *processSize;

    printf("Enter number of blocks: ");
    scanf("%d", &m);
```

```
    blockSize = (int *)malloc(m * sizeof(int));
    if (blockSize == NULL) {
        printf("Memory Allocation failed\n");
        return 1;
    }

    printf("Enter block sizes:\n");
    for (int i = 0; i < m; i++) {
        scanf("%d", &blockSize[i]);
    }

    printf("Enter number of processes: ");
    scanf("%d", &n);

    processSize = (int *)malloc(n * sizeof(int));
    if (processSize == NULL) {
        printf("Memory Allocation failed\n");
        free(blockSize);
        return 1;
    }

    printf("Enter process sizes:\n");
    for (int i = 0; i < n; i++) {
        scanf("%d", &processSize[i]);
    }
  while (1){
    printf("\nChoose allocation method:\n");
    printf("1. First Fit\n"); printf("2. Best
    Fit\n"); printf("3. Next Fit\n"); printf("4.
    Worst Fit\n"); printf("Enter your choice:
    "); scanf("%d", &choice);



    switch (choice) {
        case 1:
            printf("\nFirst Fit Allocation:\n");
            firstFit(blockSize, m, processSize, n);
            break;
        case 2:
            printf("\nBest Fit Allocation:\n");
            bestFit(blockSize, m, processSize, n);
            break;
```

```
        case 3:
            printf("\nNext Fit Allocation:\n");
            nextFit(blockSize, m, processSize, n);
            break;
        case 4:
            printf("\nWorst Fit Allocation:\n");
            worstFit(blockSize, m, processSize, n);
            break;
        default:
            printf("Invalid choice!\n");
break;
    }
}
    free(blockSize);
    free(processSize);
    return 0;
}
```

Output :




PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
ass6_os
Enter your choice: 2
Best Fit Allocation:
Process No. Process Size Block No.
1 212 4
2 417 2
3 112 3
4 426 5
Choose allocation method:
1. First Fit
2. Best Fit
3. Next Fit
4. Worst Fit
Enter your choice: 3
Next Fit Allocation:
Process No. Process Size Block No.
1 212 2
2 417 5
3 112 2
4 426 Not Allocated
Choose allocation method:
1. First Fit
2. Best Fit
3. Next Fit
4. Worst Fit
Enter your choice: 4


PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
ass6_os
Choose allocation method:
1. First Fit
2. Best Fit
3. Next Fit
4. Worst Fit
Enter your choice: 4
Worst Fit Allocation:
Process No. Process Size Block No.
1 212 5 2 417 2 3 112 5 4
426 Not Allocated