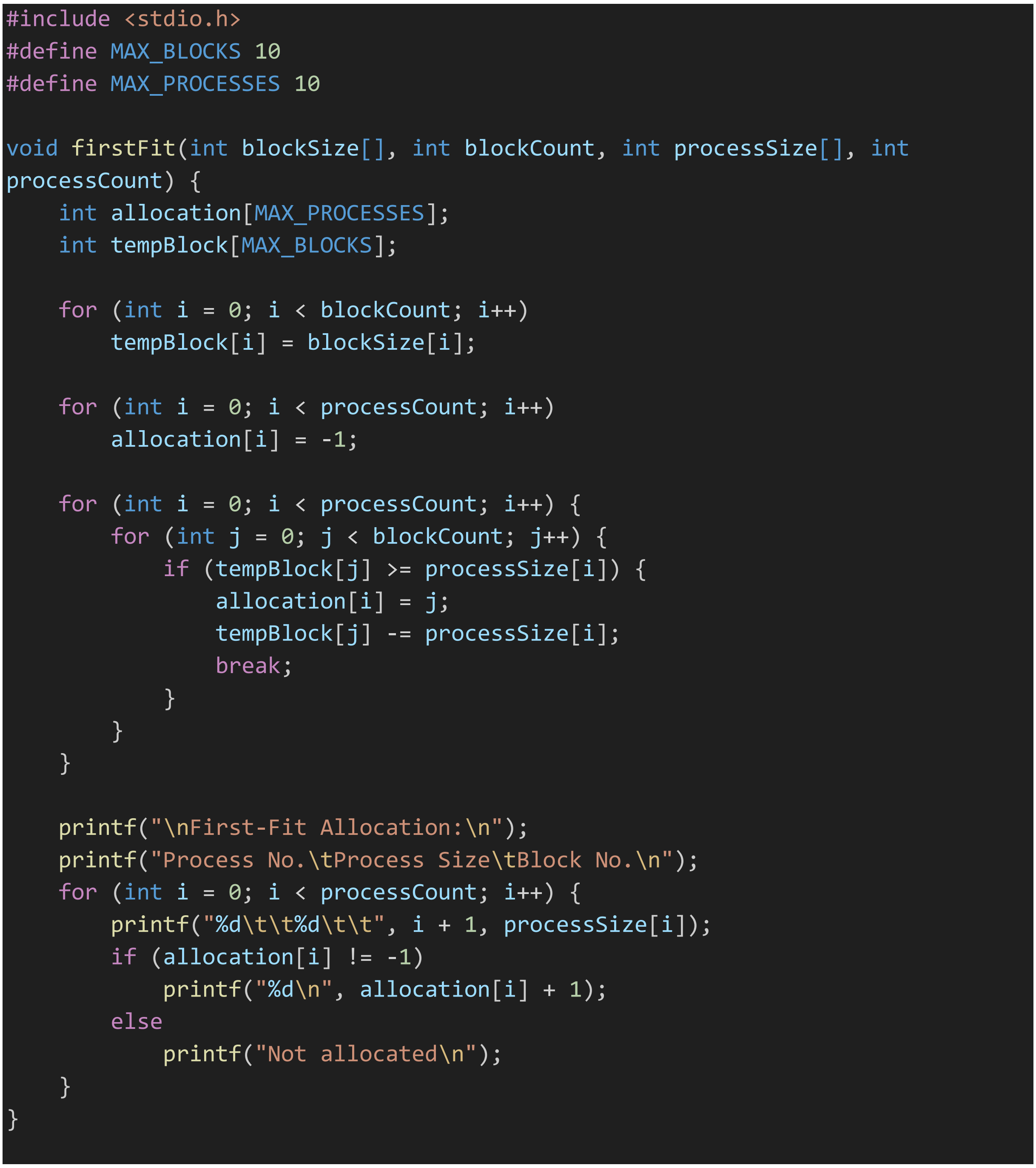
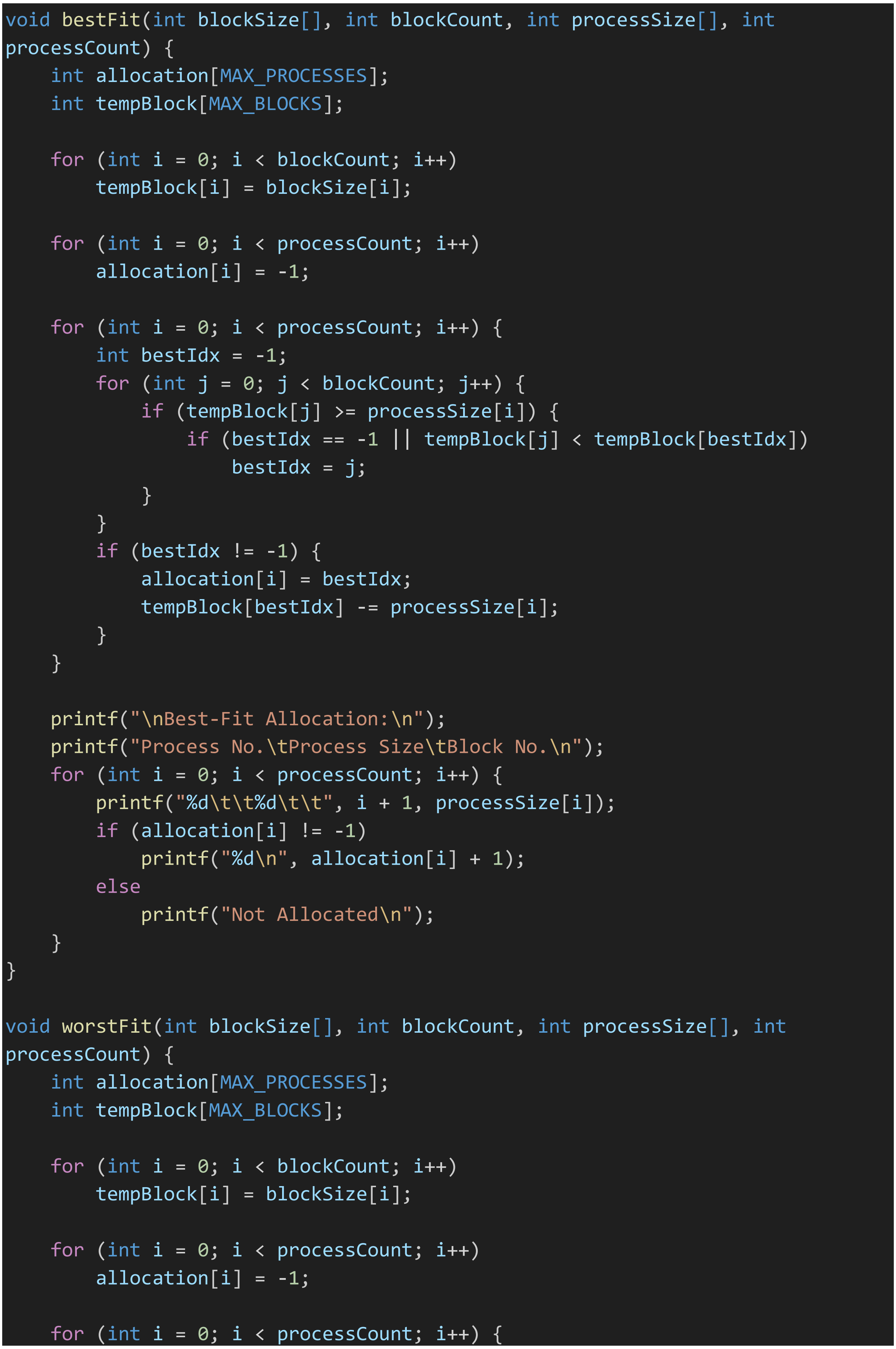
LAB 7

7) Write a C program to simulate the following contiguous memory allocation techniques.

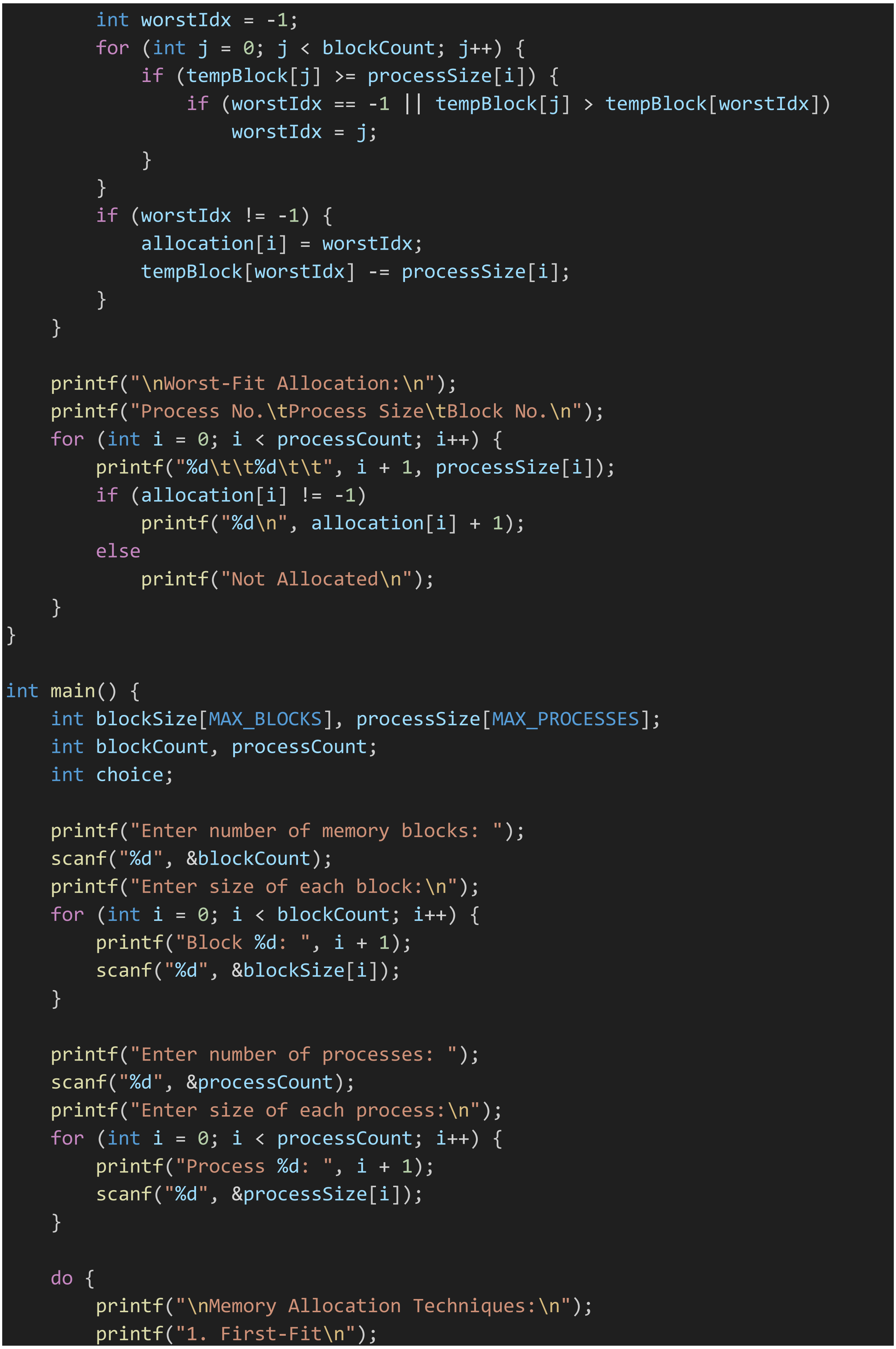
a) Worst-fit
b) Best-fit
c) First-fit



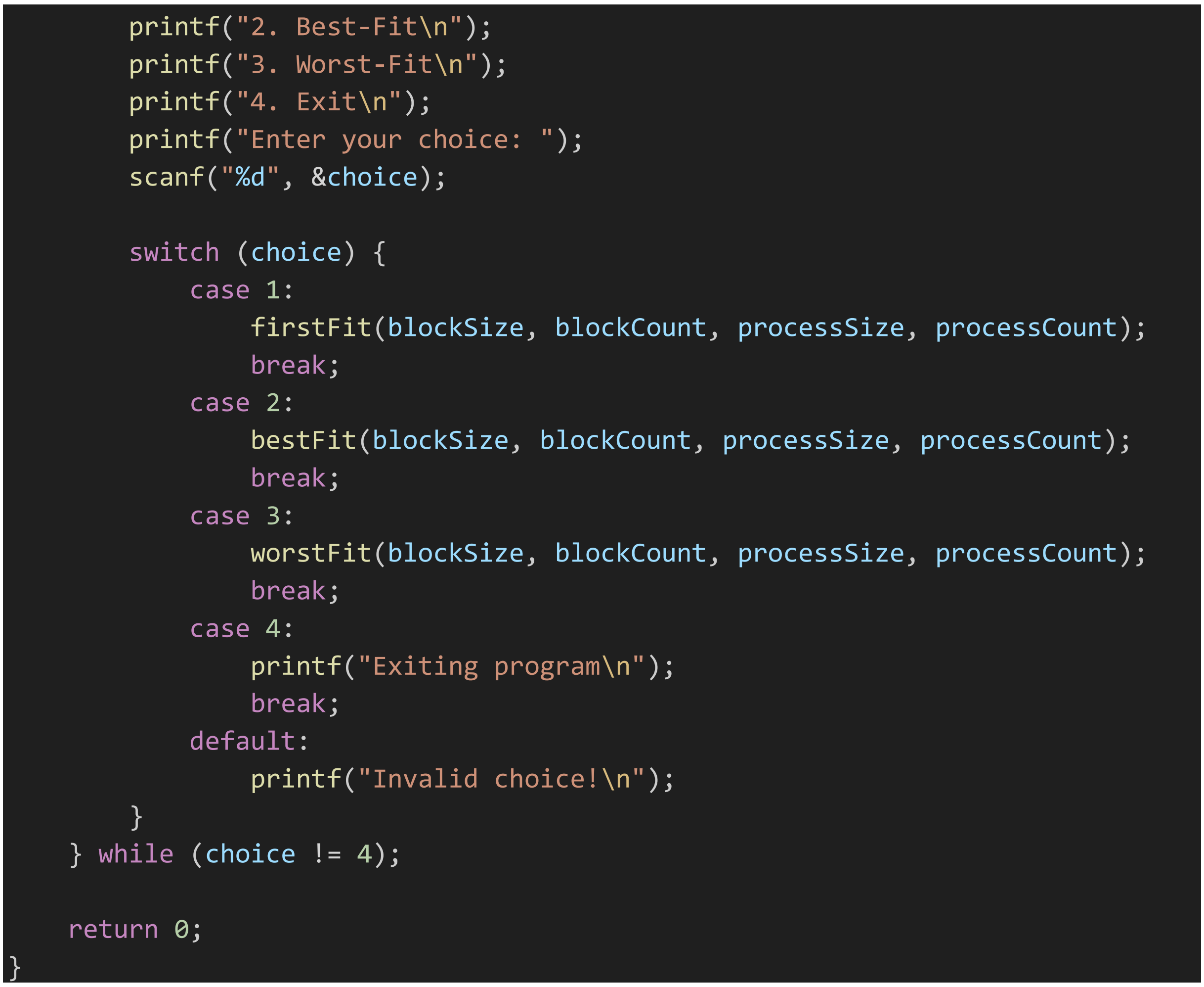
```c
#include <stdio.h>
#define MAX_BLOCKS 10
#define MAX_PROCESSES 10

void firstFit(int blockSize[], int blockCount, int processSize[], int
processCount) {
    int allocation[MAX_PROCESSES];
    int tempBlock[MAX_BLOCKS];

    for (int i = 0; i < blockCount; i++)
        tempBlock[i] = blockSize[i];

    for (int i = 0; i < processCount; i++)
        allocation[i] = -1;

    for (int i = 0; i < processCount; i++) {
        for (int j = 0; j < blockCount; j++) {
            if (tempBlock[j] >= processSize[i]) {
                allocation[i] = j;
                tempBlock[j] -= processSize[i];
                break;
            }
        }
    }

    printf("\nFirst-Fit Allocation:\n");
    printf("Process No.\tProcess Size\tBlock No.\n");
    for (int i = 0; i < processCount; 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");
    }
}
```



```c
void bestFit(int blockSize[], int blockCount, int processSize[], int
processCount) {
    int allocation[MAX_PROCESSES];
    int tempBlock[MAX_BLOCKS];

    for (int i = 0; i < blockCount; i++)
        tempBlock[i] = blockSize[i];

    for (int i = 0; i < processCount; i++)
        allocation[i] = -1;

    for (int i = 0; i < processCount; i++) {
        int bestIdx = -1;
        for (int j = 0; j < blockCount; j++) {
            if (tempBlock[j] >= processSize[i]) {
                if (bestIdx == -1 || tempBlock[j] < tempBlock[bestIdx])
                    bestIdx = j;
            }
        }
        if (bestIdx != -1) {
            allocation[i] = bestIdx;
            tempBlock[bestIdx] -= processSize[i];
        }
    }

    printf("\nBest-Fit Allocation:\n");
    printf("Process No.\tProcess Size\tBlock No.\n");
    for (int i = 0; i < processCount; 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");
    }
}

void worstFit(int blockSize[], int blockCount, int processSize[], int
processCount) {
    int allocation[MAX_PROCESSES];
    int tempBlock[MAX_BLOCKS];

    for (int i = 0; i < blockCount; i++)
        tempBlock[i] = blockSize[i];

    for (int i = 0; i < processCount; i++)
        allocation[i] = -1;

    for (int i = 0; i < processCount; i++) {
```

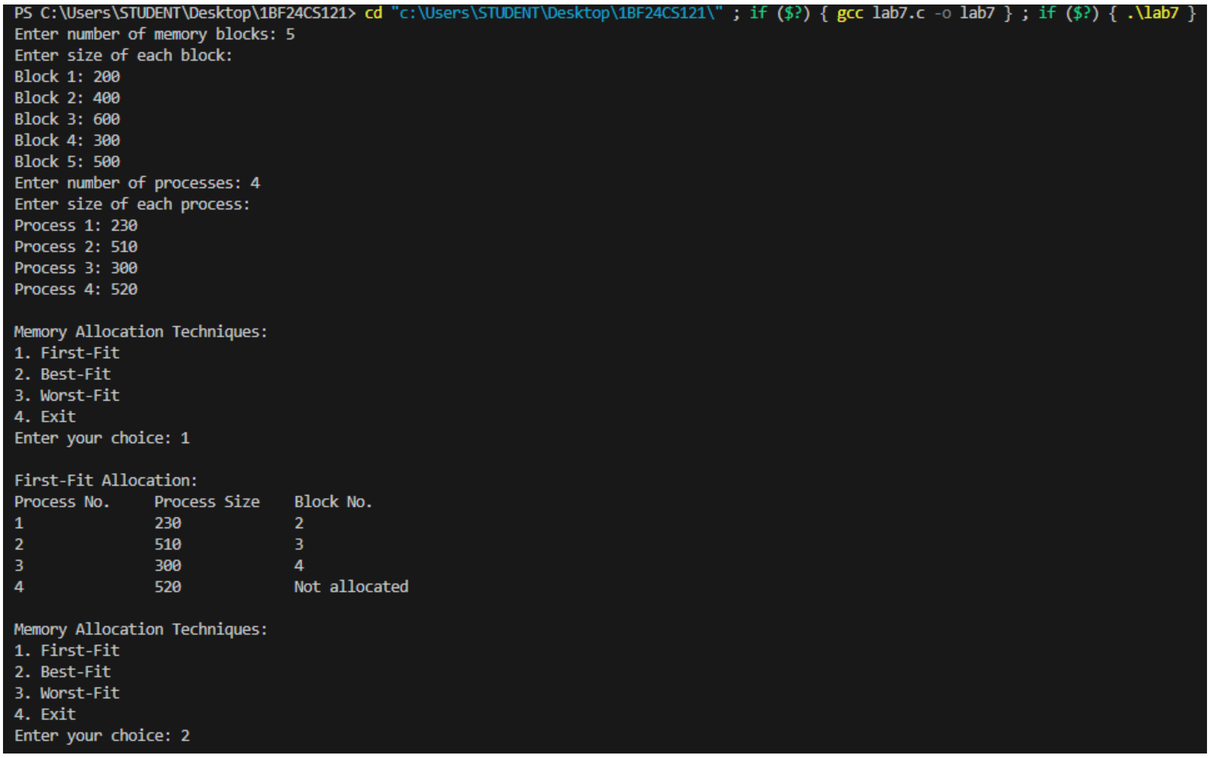


```
        int worstIdx = -1;
        for (int j = 0; j < blockCount; j++) {
            if (tempBlock[j] >= processSize[i]) {
                if (worstIdx == -1 || tempBlock[j] > tempBlock[worstIdx])
                    worstIdx = j;
            }
        }
        if (worstIdx != -1) {
            allocation[i] = worstIdx;
            tempBlock[worstIdx] -= processSize[i];
        }
    }

    printf("\nWorst-Fit Allocation:\n");
    printf("Process No.\tProcess Size\tBlock No.\n");
    for (int i = 0; i < processCount; 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[MAX_BLOCKS], processSize[MAX_PROCESSES];
    int blockCount, processCount;
    int choice;

    printf("Enter number of memory blocks: ");
    scanf("%d", &blockCount);
    printf("Enter size of each block:\n");
    for (int i = 0; i < blockCount; i++) {
        printf("Block %d: ", i + 1);
        scanf("%d", &blockSize[i]);
    }

    printf("Enter number of processes: ");
    scanf("%d", &processCount);
    printf("Enter size of each process:\n");
    for (int i = 0; i < processCount; i++) {
        printf("Process %d: ", i + 1);
        scanf("%d", &processSize[i]);
    }

    do {
        printf("\nMemory Allocation Techniques:\n");
        printf("1. First-Fit\n");
```

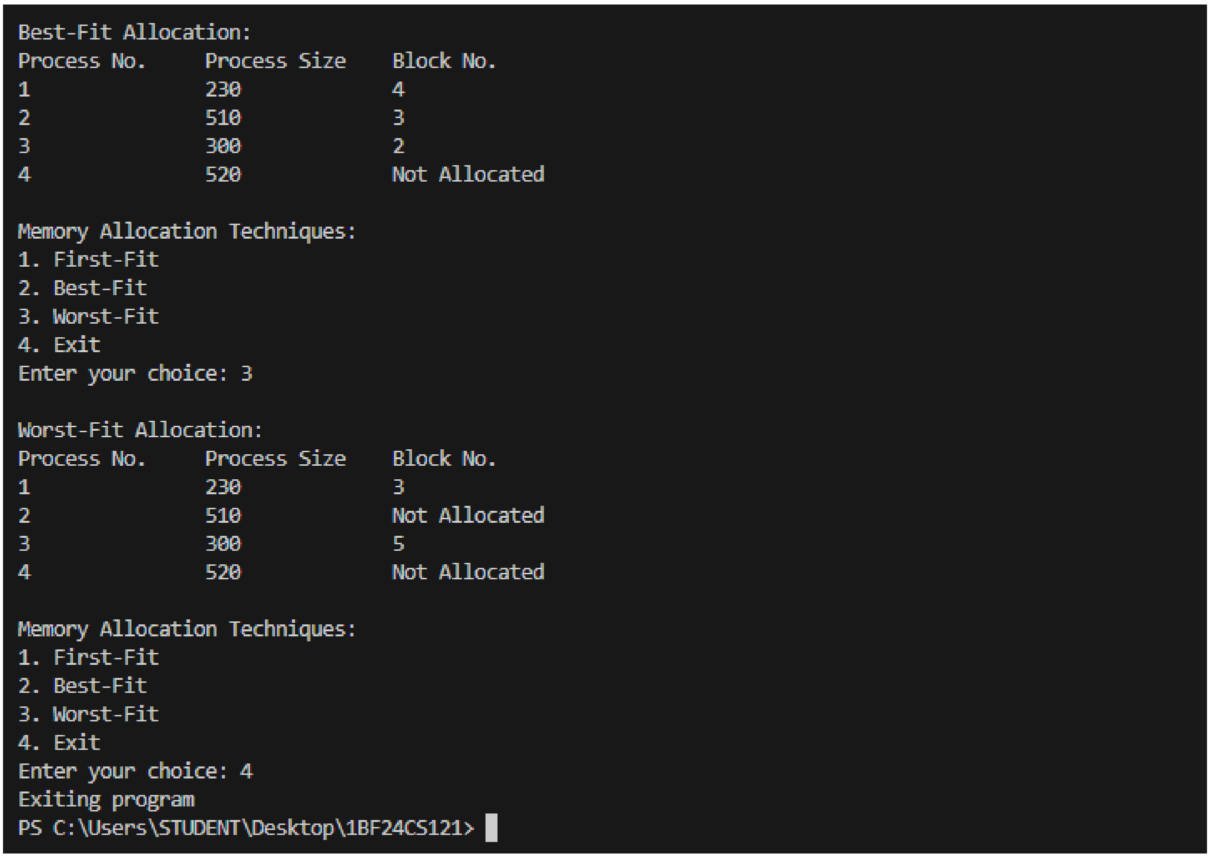


```c
        printf("2. Best-Fit\n");
        printf("3. Worst-Fit\n");
        printf("4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                firstFit(blockSize, blockCount, processSize, processCount);
                break;
            case 2:
                bestFit(blockSize, blockCount, processSize, processCount);
                break;
            case 3:
                worstFit(blockSize, blockCount, processSize, processCount);
                break;
            case 4:
                printf("Exiting program\n");
                break;
            default:
                printf("Invalid choice!\n");
        }
    } while (choice != 4);

    return 0;
}
```

OUTPUT:



```
PS C:\Users\STUDENT\Desktop\1BF24CS121> cd "c:\Users\STUDENT\Desktop\1BF24CS121\" ; if ($?) { gcc lab7.c -o lab7 } ; if ($?) { .\lab7 }
Enter number of memory blocks: 5
Enter size of each block:
Block 1: 200
Block 2: 400
Block 3: 600
Block 4: 300
Block 5: 500
Enter number of processes: 4
Enter size of each process:
Process 1: 230
Process 2: 510
Process 3: 300
Process 4: 520

Memory Allocation Techniques:
1. First-Fit
2. Best-Fit
3. Worst-Fit
4. Exit
Enter your choice: 1

First-Fit Allocation:
Process No.     Process Size    Block No.
1               230             2
2               510             3
3               300             4
4               520             Not allocated

Memory Allocation Techniques:
1. First-Fit
2. Best-Fit
3. Worst-Fit
4. Exit
Enter your choice: 2
```



```
Best-Fit Allocation:
Process No.     Process Size    Block No.
1               230             4
2               510             3
3               300             2
4               520             Not Allocated

Memory Allocation Techniques:
1. First-Fit
2. Best-Fit
3. Worst-Fit
4. Exit
Enter your choice: 3

Worst-Fit Allocation:
Process No.     Process Size    Block No.
1               230             3
2               510             Not Allocated
3               300             5
4               520             Not Allocated

Memory Allocation Techniques:
1. First-Fit
2. Best-Fit
3. Worst-Fit
4. Exit
Enter your choice: 4
Exiting program
PS C:\Users\STUDENT\Desktop\1BF24CS121> 
```