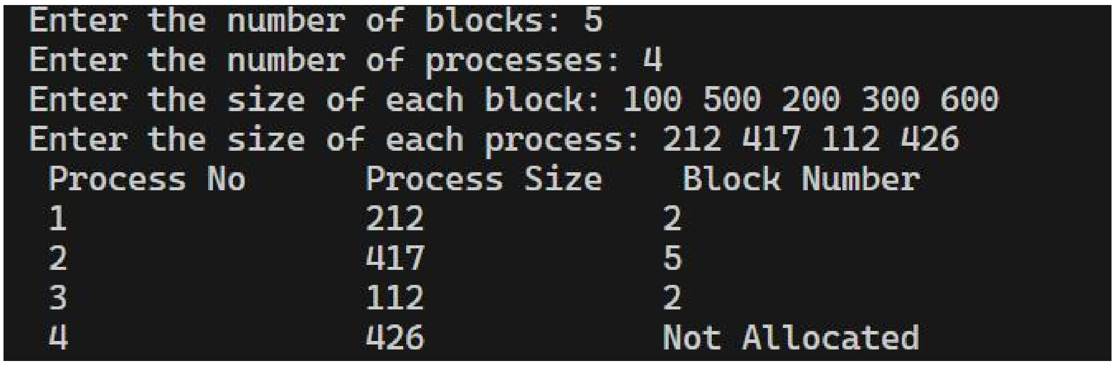
Name : Kunal Sachin Kharat
Roll no : 33
Class : CSAIML-A

Q.) Implement memory placement strategies:-

1. First Fit:

```c
#include <stdio.h>

void firstFit(int blockSize[], int m, int processSize[], int n) {
    int allocate[n]; // Initializing allocate list
    for (int i = 0; i < n; i++) {
        allocate[i] = -1;
    }
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            if (blockSize[j] >= processSize[i]) {
                allocate[i] = j;
                blockSize[j] -= processSize[i];
                break;
            }
        }
    }
    // Display the processes with the blocks that are allocated to a respective process
    printf(" Process No\t Process Size \t Block Number\n");
    for (int i = 0; i < n; i++) {
        printf(" %d\t\t %d\t\t", i + 1, processSize[i]);
        if (allocate[i] != -1) {
            printf("%d\n", allocate[i] + 1);
        } else {
            printf("Not Allocated\n");
        }
    }
}
int main() {
    int m, n;
    // get the number of blocks and processes
    printf("Enter the number of blocks: ");
    scanf("%d", &m);
    printf("Enter the number of processes: ");
    scanf("%d", &n);
    int blockSize[m], processSize[n];
    // get the size of each block
    printf("Enter the size of each block: ");
    for (int i = 0; i < m; i++) {
        scanf("%d", &blockSize[i]);
    }
    // get the size of each process
    printf("Enter the size of each process: ");
    for (int i = 0; i < n; i++) {
        scanf("%d", &processSize[i]);
    }
    // Call the function
    firstFit(blockSize, m, processSize, n);
    return 0;
}
```

Output:

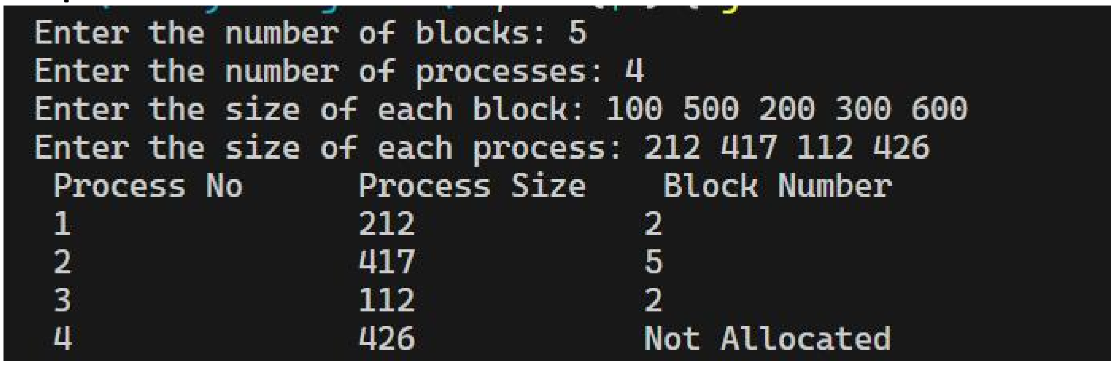


```
 Enter the number of blocks: 5
 Enter the number of processes: 4
 Enter the size of each block: 100 500 200 300 600
 Enter the size of each process: 212 417 112 426
  Process No       Process Size     Block Number
  1                212             2
  2                417             5
  3                112             2
  4                426             Not Allocated
```

2. Next Fit:

```c
#include <stdio.h>

void nextFit(int blockSize[], int m, int processSize[], int n) {
    int allocate[n]; // Initializing allocate list
    for (int i = 0; i < n; i++) {
        allocate[i] = -1;
    }
    int lastIndex = 0; // Initialize the index of the last block allocated
    for (int i = 0; i < n; i++) {
        for (int j = lastIndex; j < m; j++) {
            if (blockSize[j] >= processSize[i]) {
                allocate[i] = j;
                blockSize[j] -= processSize[i];
                lastIndex = j; // Update the index of the last block allocated
                break;
            }
        }
        // If no block is found in the remaining blocks, start searching from the
beginning

            for (int j = 0; j < lastIndex; j++) {
                if (blockSize[j] >= processSize[i]) {
                    allocate[i] = j;
                    blockSize[j] -= processSize[i];
                    lastIndex = j; // Update the index of the last block allocated
                    break;
                }
            }
    }
    // Display the processes with the blocks that are allocated to a respective process
    printf(" Process No\t Process Size \t Block Number\n");
    for (int i = 0; i < n; i++) {
        printf(" %d\t\t %d\t\t", i + 1, processSize[i]);
        if (allocate[i] != -1) {
            printf("%d\n", allocate[i] + 1);
        } else {
            printf("Not Allocated\n");
        }
    }
}
int main() {
    int m, n;
    // get the number of blocks and processes
    printf("Enter the number of blocks: ");
    scanf("%d", &m);
    printf("Enter the number of processes: ");
    scanf("%d", &n);
    int blockSize[m], processSize[n];
    // get the size of each block
    printf("Enter the size of each block: ");
    for (int i = 0; i < m; i++) {
        scanf("%d", &blockSize[i]);
    }
    // get the size of each process
    printf("Enter the size of each process: ");
    for (int i = 0; i < n; i++) {
        scanf("%d", &processSize[i]);
    }
    // Call the function
    nextFit(blockSize, m, processSize, n);
    return 0;
}
```

Output:

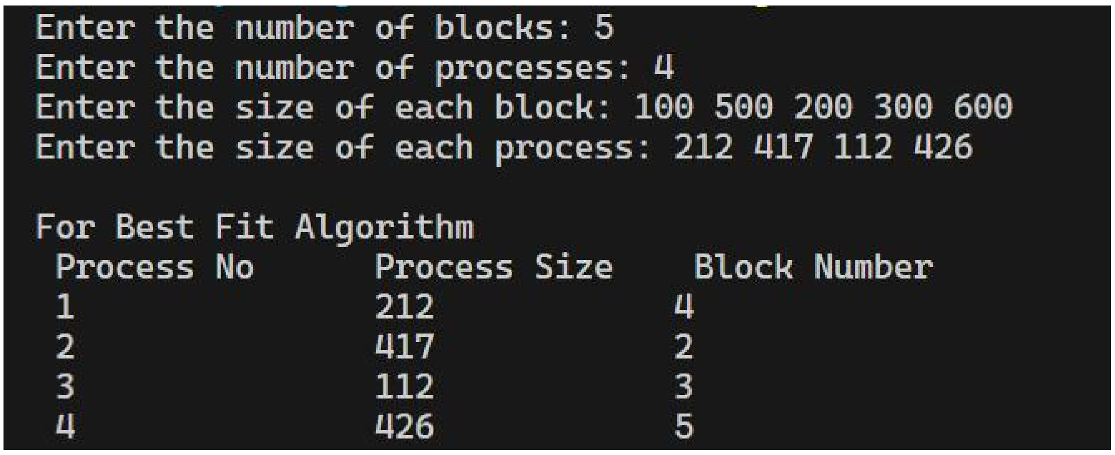


```
Enter the number of blocks: 5
Enter the number of processes: 4
Enter the size of each block: 100 500 200 300 600
Enter the size of each process: 212 417 112 426
 Process No       Process Size     Block Number
 1                212             2
 2                417             5
 3                112             2
 4                426             Not Allocated
```

3. Best Fit:

```c
#include <stdio.h>

void bestFit(int blockSize[], int m, int processSize[], int n) {
    int allocate[n]; // Initializing allocate list
    for (int i = 0; i < n; i++) {
        allocate[i] = -1;
    }
    // select the best memory block that can be allocated to a process
    for (int i = 0; i < n; i++) {
        int bestIndex = -1;
        for (int j = 0; j < m; j++) {
            if (blockSize[j] >= processSize[i]) {
                if (bestIndex == -1) {
                    bestIndex = j;
                } else if (blockSize[bestIndex] > blockSize[j]) {
                    bestIndex = j;
                }
            }
        }
        if (bestIndex != -1) {
            allocate[i] = bestIndex;
            blockSize[bestIndex] -= processSize[i];
        }
    }
    // Display the processes with the blocks that are allocated to a respective process
    printf("\nFor Best Fit Algorithm\n");
    printf(" Process No\t Process Size \t Block Number\n");
    for (int i = 0; i < n; i++) {
        printf(" %d\t\t %d\t\t", i + 1, processSize[i]);
        if (allocate[i] != -1) {
            printf("%d\n", allocate[i] + 1);
        } else {
            printf("Not Allocated\n");
        }
    }
}
int main() {
    int m, n;
    // get the number of blocks and processes
    printf("Enter the number of blocks: ");
    scanf("%d", &m);
    printf("Enter the number of processes: ");
    scanf("%d", &n);
    int blockSize[m], processSize[n];
    // get the size of each block
    printf("Enter the size of each block: ");
    for (int i = 0; i < m; i++) {
        scanf("%d", &blockSize[i]);
    }
    // get the size of each process
    printf("Enter the size of each process: ");
    for (int i = 0; i < n; i++) {
        scanf("%d", &processSize[i]);
    }
    // Call the function
    bestFit(blockSize, m, processSize, n);
    return 0;
}
```

Output:

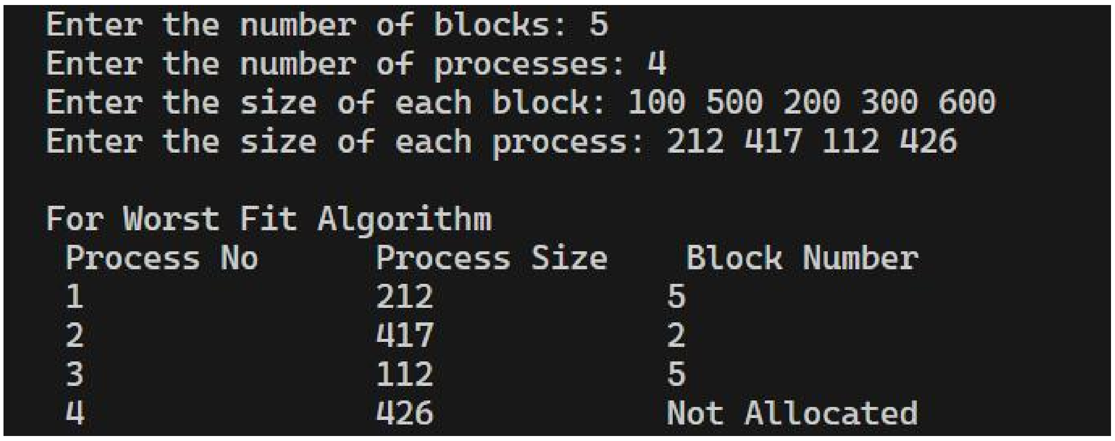


```
Enter the number of blocks: 5
Enter the number of processes: 4
Enter the size of each block: 100 500 200 300 600
Enter the size of each process: 212 417 112 426

For Best Fit Algorithm
 Process No      Process Size    Block Number
 1               212             4
 2               417             2
 3               112             3
 4               426             5
```

4. Worst Fit:

```c
#include <stdio.h>

void worstFit(int blockSize[], int m, int processSize[], int n) {
    int allocate[n]; // Initializing allocate list
    for (int i = 0; i < n; i++) {
        allocate[i] = -1;
    }
    // select the worst memory block that can be allocated to a process
    for (int i = 0; i < n; i++) {
        int worstIndex = -1;
        for (int j = 0; j < m; j++) {
            if (blockSize[j] >= processSize[i]) {
                if (worstIndex == -1) {
                    worstIndex = j;
                } else if (blockSize[worstIndex] < blockSize[j]) {
                    worstIndex = j;
                }
            }
        }
        if (worstIndex != -1) {
            allocate[i] = worstIndex;
            blockSize[worstIndex] -= processSize[i];
        }
    }
    // Display the processes with the blocks that are allocated to a respective process
    printf("\nFor Worst Fit Algorithm\n");
    printf(" Process No\t Process Size \t Block Number\n");
    for (int i = 0; i < n; i++) {
        printf(" %d\t\t %d\t\t", i + 1, processSize[i]);
        if (allocate[i] != -1) {
            printf("%d\n", allocate[i] + 1);
        } else {
            printf("Not Allocated\n");
        }
    }
}
int main() {
    int m, n;
    // get the number of blocks and processes
    printf("Enter the number of blocks: ");
    scanf("%d", &m);
    printf("Enter the number of processes: ");
    scanf("%d", &n);
    int blockSize[m], processSize[n];
    // get the size of each block
    printf("Enter the size of each block: ");
    for (int i = 0; i < m; i++) {
        scanf("%d", &blockSize[i]);
    }
    // get the size of each process
    printf("Enter the size of each process: ");
    for (int i = 0; i < n; i++) {
        scanf("%d", &processSize[i]);
    }
    // Call the function
    worstFit(blockSize, m, processSize, n);
    return 0;
}
```

Output:



```
Enter the number of blocks: 5
Enter the number of processes: 4
Enter the size of each block: 100 500 200 300 600
Enter the size of each process: 212 417 112 426

For Worst Fit Algorithm
 Process No       Process Size     Block Number
 1                212             5
 2                417             2
 3                112             5
 4                426             Not Allocated
```