NAME: MUHAMMAD SAIM NOMANI
ROLL NO: DT-22030
SUBJECT: OPERATING SYSTEM
CODE: CT-353
DATA SCIENCE
THIRD YEAR

OS LAB: 9

CODE:

```
#include <stdio.h>

int main() {
    int p[10], np, b[10], nb, ch, c[10], d[10], alloc[10], flag[10], i, j;

    printf("\nEnter the number of processes: ");
    scanf("%d", &np);

    printf("\nEnter the number of blocks: ");
    scanf("%d", &nb);

    printf("\nEnter the size of each process:");
    for (i = 0; i < np; i++) {
        printf("\nProcess %d: ", i);
        scanf("%d", &p[i]);
    }

    printf("\nEnter the block sizes:");
    for (j = 0; j < nb; j++) {
        printf("\nBlock %d: ", j);
        scanf("%d", &b[j]);
        c[j] = b[j]; // for Best Fit
        d[j] = b[j]; // for Worst Fit
    }

    if (np <= nb) {
        printf("\n1. First Fit  2. Best Fit  3. Worst Fit");

        do {
```

```
        printf("\nEnter your choice: ");
        scanf("%d", &ch);

        switch (ch) {
            case 1:
                printf("\nFirst Fit\n");
                for (i = 0; i < np; i++) {
                    flag[i] = 1;
                    for (j = 0; j < nb; j++) {
                        if (p[i] <= b[j]) {
                            alloc[j] = p[i];
                            printf("\n\nAlloc[%d]", alloc[j]);
                            printf("\n\nProcess %d of size %d is allocated in block %d of size %d", i,
p[i], j, b[j]);
                            flag[i] = 0;
                            b[j] = 0;
                            break;
                        }
                    }
                }
                for (i = 0; i < np; i++) {
                    if (flag[i] != 0)
                        printf("\n\nProcess %d of size %d is not allocated", i, p[i]);
                }
                break;

            case 2:
                printf("\nBest Fit\n");

                // Sort blocks in ascending order
                for (i = 0; i < nb; i++) {
                    for (j = i + 1; j < nb; j++) {
                        if (c[i] > c[j]) {
                            int temp = c[i];
                            c[i] = c[j];
                            c[j] = temp;
                        }
                    }
                }

                printf("\nAfter sorting block sizes:");
                for (i = 0; i < nb; i++)
                    printf("\nBlock %d: %d", i, c[i]);

                for (i = 0; i < np; i++) {
                    flag[i] = 1;
                    for (j = 0; j < nb; j++) {
                        if (p[i] <= c[j]) {
```

```
                        alloc[j] = p[i];
                        printf("\n\nAlloc[%d]", alloc[j]);
                        printf("\n\nProcess %d of size %d is allocated in block %d of size %d", i,
p[i], j, c[j]);
                        flag[i] = 0;
                        c[j] = 0;
                        break;
                    }
                }
            }

            for (i = 0; i < np; i++) {
                if (flag[i] != 0)
                    printf("\n\nProcess %d of size %d is not allocated", i, p[i]);
            }
            break;

        case 3:
            printf("\nWorst Fit\n");

            // Sort blocks in descending order
            for (i = 0; i < nb; i++) {
                for (j = i + 1; j < nb; j++) {
                    if (d[i] < d[j]) {
                        int temp = d[i];
                        d[i] = d[j];
                        d[j] = temp;
                    }
                }
            }

            printf("\nAfter sorting block sizes:");
            for (i = 0; i < nb; i++)
                printf("\nBlock %d: %d", i, d[i]);

            for (i = 0; i < np; i++) {
                flag[i] = 1;
                for (j = 0; j < nb; j++) {
                    if (p[i] <= d[j]) {
                        alloc[j] = p[i];
                        printf("\n\nAlloc[%d]", alloc[j]);
                        printf("\n\nProcess %d of size %d is allocated in block %d of size %d", i,
p[i], j, d[j]);
                        flag[i] = 0;
                        d[j] = 0;
                        break;
                    }
                }
```

```
                }

                for (i = 0; i < np; i++) {
                    if (flag[i] != 0)
                        printf("\n\nProcess %d of size %d is not allocated", i, p[i]);
                }
                break;

            default:
                printf("Invalid Choice…!");
                break;
        }

    } while (ch <= 3);
  }

  return 0;
}
```

OUTPUT:

```
Enter the number of processes: 4
Enter the number of memory blocks: 5
Enter the size of each process:
Process 0: 212
Process 1: 417
Process 2: 112
Process 3: 426
Enter the size of each memory block:
Block 0: 100
Block 1: 500
Block 2: 200
Block 3: 300
Block 4: 600

1. First Fit  2. Best Fit  3. Worst Fit

Enter your choice: 1

First Fit:

Process 0 of size 212 is allocated in block 1 of size 500
Process 1 of size 417 is allocated in block 4 of size 600
Process 2 of size 112 is allocated in block 2 of size 200
Process 3 of size 426 is not allocated
Enter your choice: 2
Best Fit:
After sorting block sizes:
Block 0: 100
Block 1: 200
Block 2: 300
Block 3: 500
Block 4: 600

Process 0 of size 212 is allocated in block 2 of size 300
Process 1 of size 417 is allocated in block 3 of size 500
Process 2 of size 112 is allocated in block 1 of size 200
Process 3 of size 426 is allocated in block 4 of size 600
Enter your choice: 3

Worst Fit:
After sorting block sizes:
Block 0: 600
Block 1: 500
Block 2: 300
Block 3: 200
Block 4: 100

Process 0 of size 212 is allocated in block 0 of size 600
Process 1 of size 417 is allocated in block 1 of size 500
Process 2 of size 112 is allocated in block 2 of size 300
Process 3 of size 426 is not allocated
```