ASHNA SHAIKH [DT-22019]

OPERATING SYSTEM LAB [CT-353]

LAB 09

CODE:

```c
#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:\n");
    for (i = 0; i < np; i++) {
        printf("Process %d: ", i);
        scanf("%d", &p[i]);
    }

    printf("\nEnter the block sizes:\n");
    for (j = 0; j < nb; j++) {
        printf("Block %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\n2. Best Fit\n3. Worst Fit\n");

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

            switch (ch) {
                case 1:  // First Fit
                    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("\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;
```

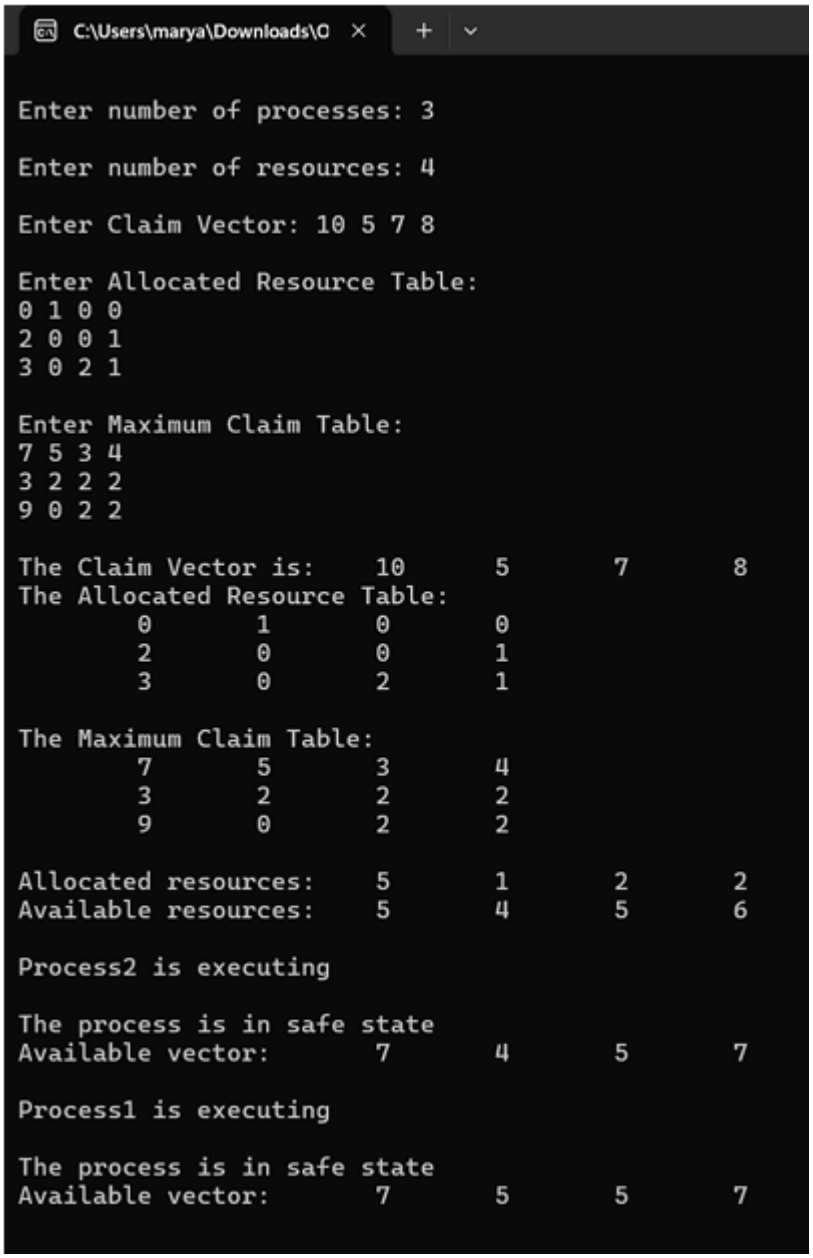
```
                }
            }
        }
        for (i = 0; i < np; i++) {
            if (flag[i] != 0)
                printf("\nProcess %d of size %d is not allocated", i, p[i]);
        }
        break;

    case 2:  // Best Fit
        printf("\nBest Fit\n");
        // Sort block sizes 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:\n");
        for (i = 0; i < nb; i++)
            printf("Block %d: %d\n", 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("\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("\nProcess %d of size %d is not allocated", i, p[i]);
        }
        break;
```

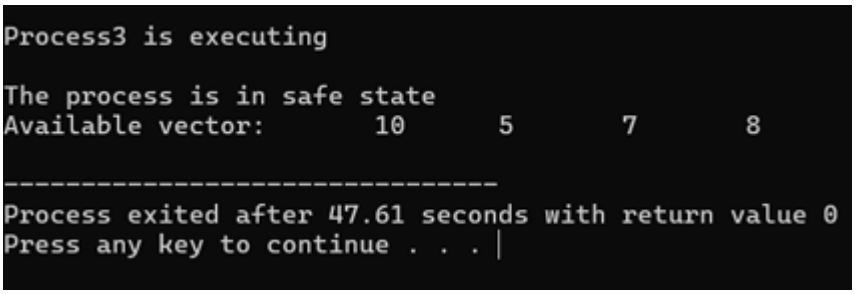
```
            case 3:  // Worst Fit
                printf("\nWorst Fit\n");
                // Sort block sizes in descending order
                for (i = 0; i < nb; i++) {
                    for (j = i + 1; j < nb; j++) {
                        if (d[i] < d[j]) {
                            int temp = d[i];
                            d[i] = d[j];
                            d[j] = temp;
                        }
                    }
                }

                printf("\nAfter sorting block sizes:\n");
                for (i = 0; i < nb; i++)
                    printf("Block %d: %d\n", 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("\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("\nProcess %d of size %d is not allocated", i, p[i]);
                }
                break;

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

    } while (ch <= 3);
}
```

OUTPUT:

```
Enter the number of processes: 4

Enter the number of blocks: 5

Enter the size of each process:
Process 0: 212
Process 1: 417
Process 2: 112
Process 3: 426

Enter the block sizes:
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
```

```
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
Enter your choice:
```


C:\Users\marya\Downloads\O
Enter number of processes: 3
Enter number of resources: 4
Enter Claim Vector: 10 5 7 8
Enter Allocated Resource Table:
0 1 0 0
2 0 0 1
3 0 2 1
Enter Maximum Claim Table:
7 5 3 4
3 2 2 2
9 0 2 2
The Claim Vector is: 10 5 7 8
The Allocated Resource Table:
0 1 0 0
2 0 0 1
3 0 2 1
The Maximum Claim Table:
7 5 3 4
3 2 2 2
9 0 2 2
Allocated resources: 5 1 2 2
Available resources: 5 4 5 6
Process2 is executing
The process is in safe state
Available vector: 7 4 5 7
Process1 is executing
The process is in safe state
Available vector: 7 5 5 7


Process3 is executing
The process is in safe state
Available vector: 10 5 7 8
Process exited after 47.61 seconds with return value 0
Press any key to continue . . .