

Course Name: Operating systems

LAB: 09

Submitted By: Ezaan Khan

Roll: DT-22046

Program:

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

```
int main() {
```

```
    int p[10], np, b[10], nb, ch;
```

```
    int c[10], d[10], alloc[10], flag[10], i, j;
```

```
    printf("\nEnter the number of processes: ");
```

```
    scanf("%d", &np);
```

```
    printf("Enter the number of memory blocks: ");
```

```
    scanf("%d", &nb);
```

```
    printf("Enter the size of each process:\n");
```

```
    for (i = 0; i < np; i++) {
```

```
        printf("Process %d: ", i);
```

```
        scanf("%d", &p[i]);
```

```
    }
```

```
    printf("Enter the size of each block:\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) {

    do {

        printf("\n1. First Fit\n2. Best Fit\n3. Worst Fit\n4. Exit\nEnter your choice: ");

        scanf("%d", &ch);

        for (i = 0; i < np; i++) flag[i] = 1; // reset flags

        for (j = 0; j < nb; j++) {

            c[j] = b[j]; // reset for best fit

            d[j] = b[j]; // reset for worst fit

        }

        switch (ch) {

            case 1:

                printf("\n--- First Fit ---\n");

                for (i = 0; i < np; i++) {

                    for (j = 0; j < nb; j++) {

                        if (p[i] <= b[j]) {

                            alloc[i] = j;

                            printf("Process %d of size %d allocated in block %d of size %d\n", i, p[i], j, b[j]);

                            flag[i] = 0;

                            b[j] = 0;

```

```

        break;

    }

}

if (flag[i]) {

    printf("Process %d of size %d is not allocated\n", i, p[i]);

}

}

break;

```

case 2:

```

printf("\n--- Best Fit ---\n");

// sort blocks in ascending order

for (i = 0; i < nb - 1; i++) {

    for (j = i + 1; j < nb; j++) {

        if (c[i] > c[j]) {

            int temp = c[i];

            c[i] = c[j];

            c[j] = temp;

        }

    }

}

for (i = 0; i < np; i++) {

    for (j = 0; j < nb; j++) {

        if (p[i] <= c[j]) {

            alloc[i] = j;

            printf("Process %d of size %d allocated in block %d of size %d\n", i, p[i], j, c[j]);

            flag[i] = 0;

```

```

        c[j] = 0;

        break;

    }

}

if (flag[i]) {

    printf("Process %d of size %d is not allocated\n", i, p[i]);

}

}

break;

```

case 3:

```

printf("\n--- Worst Fit ---\n");

// sort blocks in descending order

for (i = 0; i < nb - 1; i++) {

    for (j = i + 1; j < nb; j++) {

        if (d[i] < d[j]) {

            int temp = d[i];

            d[i] = d[j];

            d[j] = temp;

        }

    }

}

for (i = 0; i < np; i++) {

    for (j = 0; j < nb; j++) {

        if (p[i] <= d[j]) {

            alloc[i] = j;

            printf("Process %d of size %d allocated in block %d of size %d\n", i, p[i], j, d[j]);

```

```

        flag[i] = 0;

        d[j] = 0;

        break;

    }

}

if (flag[i]) {

    printf("Process %d of size %d is not allocated\n", i, p[i]);

}

}

break;

```

case 4:

```

    printf("Exiting...\n");

    break;

```

default:

```

    printf("Invalid choice. Try again.\n");

}

```

```

} while (ch != 4);

```

```

} else {

```

```

    printf("Number of processes should be less than or equal to number of blocks.\n");

```

```

}

```

```

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 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
4. Exit
Enter your choice: 1
```

```
--- First Fit ---
Process 0 of size 212 allocated in block 1 of size 500
Process 1 of size 417 allocated in block 4 of size 600
Process 2 of size 112 allocated in block 2 of size 200
Process 3 of size 426 is not allocated
```

```
1. First Fit
2. Best Fit
3. Worst Fit
4. Exit
```

Enter your choice: 2

--- Best Fit ---

Process 0 of size 212 allocated in block 4 of size 300

Process 1 of size 417 is not allocated

Process 2 of size 112 is not allocated

Process 3 of size 426 is not allocated

1. First Fit

2. Best Fit

3. Worst Fit

4. Exit

Enter your choice: 3

--- Worst Fit ---

Process 0 of size 212 allocated in block 0 of size 300

Process 1 of size 417 is not allocated

Process 2 of size 112 is not allocated

Process 3 of size 426 is not allocated

1. First Fit

2. Best Fit

3. Worst Fit

4. Exit

Enter your choice: |