

Operating Systems

CT-353

Name: Sabrina Shahzad

Roll No. : DT-026

Lab 08:

MEMORY ALLOCATION TECHNIQUES

```
#include <stdio.h>

int main() {

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

    for(i = 0; i < 10; i++) {

        flag[i] = 1; // Process not allocated

        alloc[i] = -1; // Block not allocated

    }

    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]; // Create a copy for Best Fit

        d[j] = b[j]; // Create a copy 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++) {
                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; // Block is now full
                        break;
                    }
                }
            }
            if(flag[i] == 1) {
                printf("\nProcess %d of size %d is not allocated", i, p[i]);
            }
        }
        Break;
        case 2: // Best Fit
            printf("\nBest Fit\n");
            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++) {
    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; // Block is now full
            break;
        }
    }
    if(flag[i] == 1) {
        printf("\nProcess %d of size %d is not allocated", i, p[i]);
    }
}
Break;

case 3: // Worst Fit
printf("\nWorst Fit\n");
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++) {
        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; // Block is now full

                break;
            }
        }

        if(flag[i] == 1) {
            printf("\nProcess %d of size %d is not allocated", i, p[i]);
        }
    }

    break;

default:
    printf("Invalid Choice...\n");
}

} while(ch <= 3);

} else {

    printf("\nNumber of processes cannot be greater than the number of blocks.\n");

}

return 0;

}

```

Output:

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

Enter your choice: 1

First Fit

Process 0 of size 100 is allocated in block 0 of size 500
Process 1 of size 200 is allocated in block 1 of size 300
Process 2 of size 300 is allocated in block 4 of size 400
Process 3 of size 400 is not allocated
Enter your choice: 2

Best Fit

After sorting block sizes:
Block 0: 100
Block 1: 200
Block 2: 300
Block 3: 400
Block 4: 500

Process 0 of size 100 is allocated in block 0 of size 100
Process 1 of size 200 is allocated in block 1 of size 200
Process 2 of size 300 is allocated in block 2 of size 300
Process 3 of size 400 is allocated in block 3 of size 400
Enter your choice: 3

Worst Fit

After sorting block sizes:
Block 0: 500
Block 1: 400
Block 2: 300
Block 3: 200
Block 4: 100

Process 0 of size 100 is allocated in block 0 of size 500
Process 1 of size 200 is allocated in block 1 of size 400
Process 2 of size 300 is allocated in block 2 of size 300
Enter your choice:
```