

Name: K.SaiKrishna

Reg-No: 192311106

21.Develop a C program to implement the worst fit algorithm of memory management.

Aim:

To implement the **Worst Fit** memory allocation algorithm in C for managing memory allocation to processes, ensuring the largest free memory block is allocated to a process.

Algorithm:

1. Input the size of memory blocks and processes.
2. For each process:
 - o Find the largest memory block that can fit the process.
 - o Allocate the block to the process.
 - o Reduce the block's size by the process's size.
3. If no suitable block is found, the process remains unallocated.
4. Display the allocation results.

Procedure:

1. Take input for the sizes of memory blocks and processes.
2. Traverse the memory blocks to find the largest block for each process.
3. Update memory block size and allocation details.
4. Print the allocation results for each process.

Code:

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

```
int main() {
```

```
    int blocks[10], processes[10], allocation[10];
```

```
    int nBlocks, nProcesses;
```

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

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

```
    printf("Enter sizes of memory blocks: ");
```

```
for (int i = 0; i < nBlocks; i++) scanf("%d", &blocks[i]);
```

```
printf("Enter number of processes: ");
```

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

```
printf("Enter sizes of processes: ");
```

```
for (int i = 0; i < nProcesses; i++) {
```

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

```
    allocation[i] = -1;
```

```
}
```

```
for (int i = 0; i < nProcesses; i++) {
```

```
    int worstIdx = -1;
```

```
    for (int j = 0; j < nBlocks; j++) {
```

```
        if (blocks[j] >= processes[i]) {
```

```
            if (worstIdx == -1 || blocks[j] > blocks[worstIdx])
```

```
                worstIdx = j;
```

```
        }
```

```
    }
```

```
    if (worstIdx != -1) {
```

```
        allocation[i] = worstIdx;
```

```
        blocks[worstIdx] -= processes[i];
```

```
    }
```

```
}
```

```

printf("\nProcess No.\tProcess Size\tBlock No.\n");

for (int i = 0; i < nProcesses; i++) {

    printf("%d\t%d\t", i + 1, processes[i]);

    if (allocation[i] != -1)

        printf("%d\n", allocation[i] + 1);

    else

        printf("Not Allocated\n");

}

return 0;

}

```

Output:

The screenshot shows a C++ IDE with a dark theme. On the left is a sidebar with a user profile 'Welcome, K Sai Krishna' and navigation links: 'Create New Project', 'My Projects', 'Classroom' (marked as 'new'), 'Learn Programming', 'Programming Questions', 'Upgrade', and 'Logout'. The main editor displays the C++ code for the worstFit algorithm. The code includes comments in orange and uses standard C++ syntax. The output window at the bottom shows the results of the program execution.

```

2 void worstFit(int blockSize[], int blocks, int processSize[], int pro
3     int allocation[processes];
4
5
6     // Initialize all allocations to -1 (unallocated)
7     for (int i = 0; i < processes; i++) {
8         allocation[i] = -1;
9     }
10
11     // Allocate memory to processes
12     for (int i = 0; i < processes; i++) {
13         int worstIndex = -1;
14
15         for (int j = 0; j < blocks; j++) {
16             if (blockSize[j] >= processSize[i]) {
17                 if (worstIndex == -1 || blockSize[j] > blockSize[wors
18                     worstIndex = j;
19             }
20         }
21     }
22
23     // If a suitable block is found
24     if (worstIndex != -1) {
25         allocation[i] = worstIndex;
26         blockSize[worstIndex] -= processSize[i];
27     }
28 }
29

```

The output window displays the following table:

Process No.	Process Size	Block No.
1	212	5
2	417	2
3	112	5
4	426	Not Allocated

Below the table, the output window shows the message: "...Program finished with exit code 0" and "Press ENTER to exit console."

Result:

his demonstrates the **Worst Fit** algorithm where the process is allocated the largest block that fits, or remains unallocated if no suitable block is available.