# Experiment 5
# First Fit, Best Fit and Worst Fit

**Name:-Marwin Shroff**
**SAP:60004200097**
**Sub:Operating Systems**

**AIM**: To implement various memory allocation techniques like first fit, best fit and worst fit.

**THEORY:**

• **FIRST FIT:** This method keeps the free/busy list of jobs organized by memory location, low-ordered to high-ordered memory. In this method, first job claims the first available memory space more than or equal to its size. The operating system doesn't search for appropriate partition but just allocate the job to the nearest memory partition available with sufficient size.

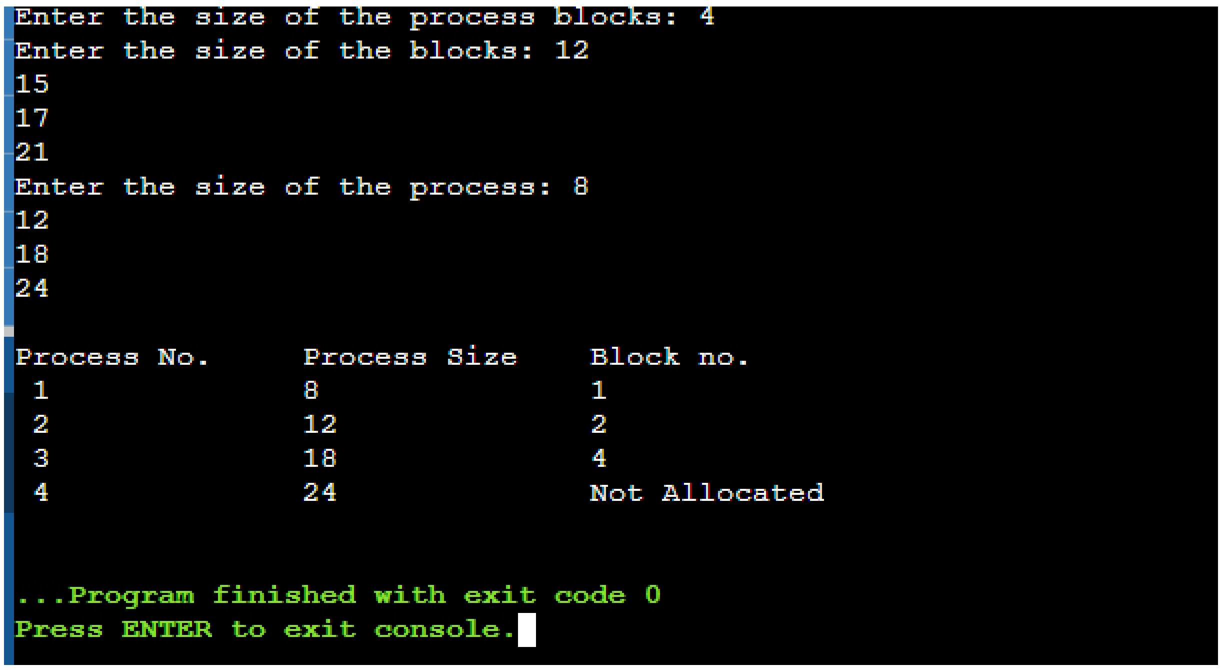
• **BEST FIT:** This method keeps the free/busy list in order by size-smallest to largest. In this method, the operating system first searches the whole of the memory according to the size of the given job and allocates it to the closest-fitting free partition in the memory, making it able to use memory efficiently. Here the jobs are in the order from smallest job to largest job.

• **WORST FIT**: In this allocation technique, the process traverses the whole memory and always search for the largest hole/partition, and then the process is placed in that hole/partition. It is a slow process because it has to traverse the entire memory to search the largest hole.

**Code:**

• **FIRST FIT:**

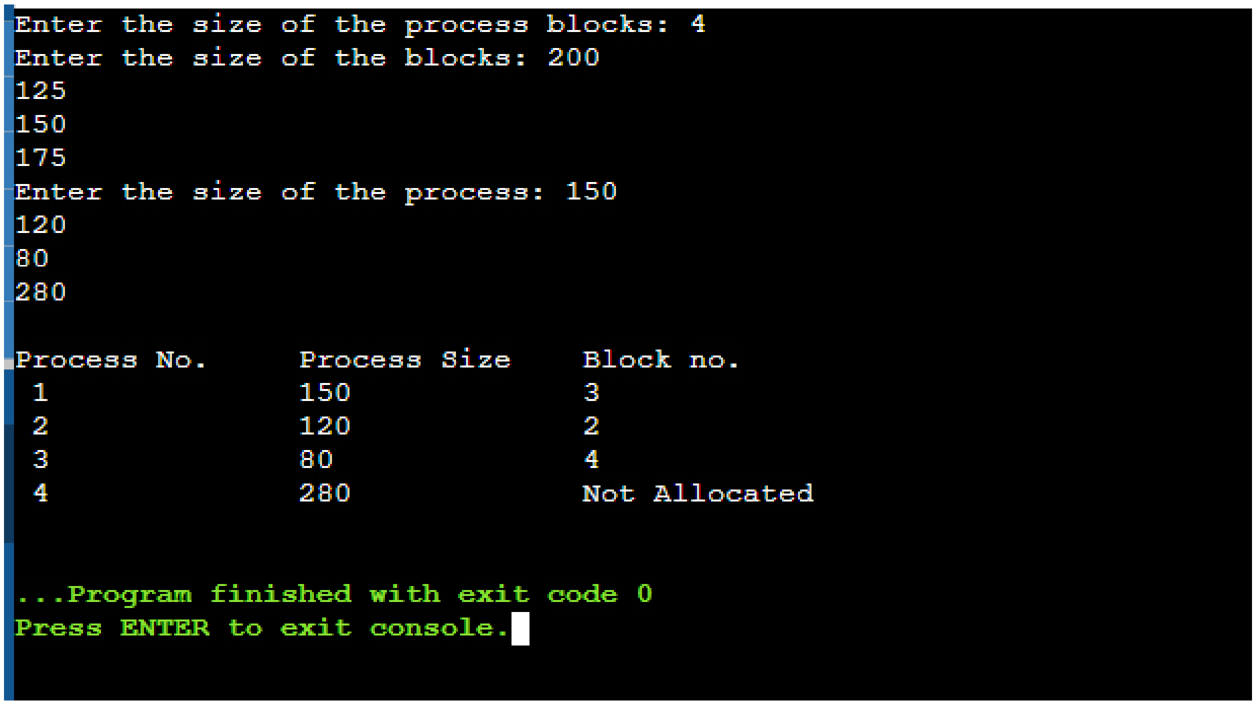
```
#include<bits/stdc++.h>
using namespace std;
void First_Fit(int block_size[], int total_blocks, int process_size[], int
total_process) {
 int allocation[total_process];
 memset(allocation, -1, sizeof(allocation));
 for (int i = 0; i < total_process; i++) {
 for (int j = 0; j < total_blocks; j++) {
 if (block_size[j] >= process_size[i]) {
 allocation[i] = j;
 block_size[j] -= process_size[i];
 break;
 }
 }
 }
 cout << "\nProcess No.\tProcess Size\tBlock no.\n";
 for (int i = 0; i < total_process; i++) {
 cout << " " << i+1 << "\t\t" << process_size[i] << "\t\t";
 if (allocation[i] != -1)
 cout << allocation[i] + 1;
 else
 cout << "Not Allocated";
 cout << endl;
 }
}
int main() {
 int n;
 printf("Enter the size of the process blocks: ");
 scanf("%d", &n);
 int block_size[n];
 printf("Enter the size of the blocks: ");
 for(int i = 0; i<n; i++){
 scanf("%d", &block_size[i]);
 }
 int process_size[n];
 printf("Enter the size of the process: ");
 for(int i = 0; i<n; i++){
 scanf("%d", &process_size[i]);
 }
 int total_blocks = sizeof(block_size) / sizeof(block_size[0]);
 int total_process = sizeof(process_size) / sizeof(process_size[0]);
 First_Fit(block_size, total_blocks, process_size, total_process);
 return 0 ;
}
```



```
Enter the size of the process blocks: 4
Enter the size of the blocks: 12
15
17
21
Enter the size of the process: 8
12
18
24

Process No.      Process Size     Block no.
 1               8                1
 2               12               2
 3               18               4
 4               24               Not Allocated


...Program finished with exit code 0
Press ENTER to exit console.
```

• **BEST FIT:**

```
#include<bits/stdc++.h>
using namespace std;
void bestfit(int bsize[], int m, int psize[], int n) {
 int alloc[n];
 memset(alloc, -1, sizeof(alloc));
 for (int i=0; i<n; i++) {
 int bestIdx = -1;
 for (int j=0; j<m; j++) {
 if (bsize[j] >= psize[i]) {
 if (bestIdx == -1)
 bestIdx = j;
 else if (bsize[bestIdx] > bsize[j])
 bestIdx = j;
 }
 }
 if (bestIdx != -1) {
 alloc[i] = bestIdx;
 bsize[bestIdx] -= psize[i];
 }
 }
 cout << "\nProcess No.\tProcess Size\tBlock no.\n";
 for (int i = 0; i < n; i++) {
 cout << " " << i+1 << "\t\t" << psize[i] << "\t\t";
 if (alloc[i] != -1)
 cout << alloc[i] + 1;
 else
 cout << "Not Allocated";
 cout << endl;
 }
}
int main() {
 int a;
 printf("Enter the size of the process blocks: ");
 scanf("%d", &a);
 int bsize[a];
 printf("Enter the size of the blocks: ");
 for(int i = 0; i<a; i++){
 scanf("%d", &bsize[i]);
 }
 int psize[a];
 printf("Enter the size of the process: ");
 for(int i = 0; i<a; i++){
 scanf("%d", &psize[i]);
 }
 int m = sizeof(bsize)/sizeof(bsize[0]);
 int n = sizeof(psize)/sizeof(psize[0]);
 bestfit(bsize, m, psize, n);
 return 0 ;
}
```

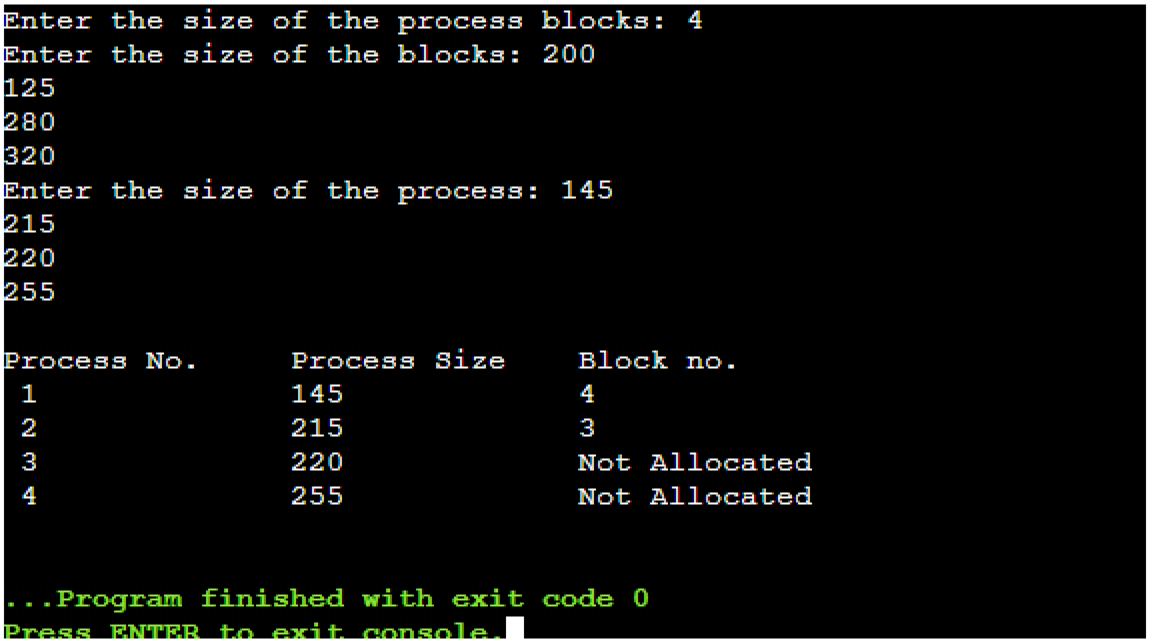


```
Enter the size of the process blocks: 4
Enter the size of the blocks: 200
125
150
175
Enter the size of the process: 150
120
80
280

Process No.      Process Size    Block no.
 1               150             3
 2               120             2
 3               80              4
 4               280             Not Allocated


...Program finished with exit code 0
Press ENTER to exit console.
```

• **WORST FIT:**

```
#include<bits/stdc++.h>
using namespace std;
void worstFit(int blockSize[], int m, int processSize[], int n){
int allocation[n];
memset(allocation, -1, sizeof(allocation));
for (int i=0; i<n; i++){
int wstIdx = -1;
for (int j=0; j<m; j++){
if (blockSize[j] >= processSize[i]){
if (wstIdx == -1)
wstIdx = j;
else if (blockSize[wstIdx] < blockSize[j])
wstIdx = j;
}
}
if (wstIdx != -1){
```

```
allocation[i] = wstIdx;
blockSize[wstIdx] -= processSize[i];
}
}
cout << "\nProcess No.\tProcess Size\tBlock no.\n";
for (int i = 0; i < n; i++){
cout << " " << i+1 << "\t\t" << processSize[i] << "\t\t";
if (allocation[i] != -1)
cout << allocation[i] + 1;
else
cout << "Not Allocated";
cout << endl;
}
}
int main(){
int a;
 printf("Enter the size of the process blocks: ");
 scanf("%d", &a);
 int blockSize[a];
 printf("Enter the size of the blocks: ");
 for(int i = 0; i<a; i++){
 scanf("%d", &blockSize[i]);
 }
 int processSize[a];
 printf("Enter the size of the process: ");
 for(int i = 0; i<a; i++){
 scanf("%d", &processSize[i]);
 }
int m = sizeof(blockSize)/sizeof(blockSize[0]);
int n = sizeof(processSize)/sizeof(processSize[0]);
worstFit(blockSize, m, processSize, n);
return 0 ;
}
```



```
Enter the size of the process blocks: 4
Enter the size of the blocks: 200
125
280
320
Enter the size of the process: 145
215
220
255

Process No.     Process Size    Block no.
 1              145             4
 2              215             3
 3              220             Not Allocated
 4              255             Not Allocated


...Program finished with exit code 0
Press ENTER to exit console.
```