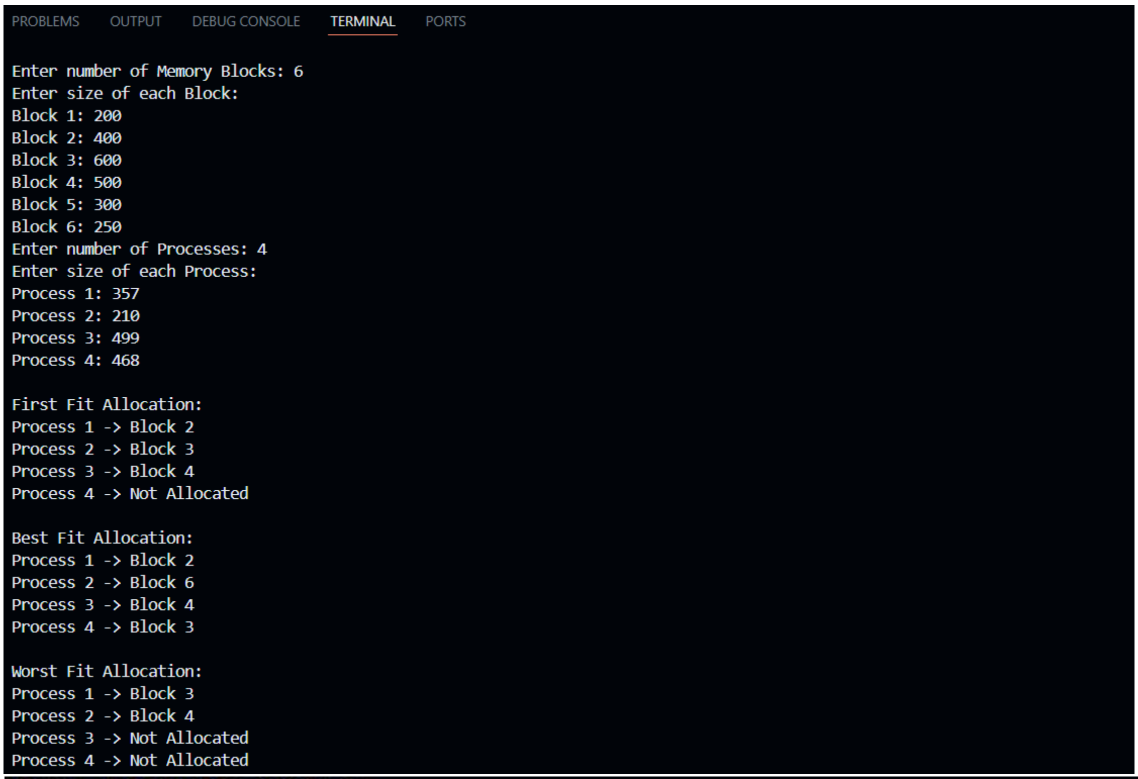
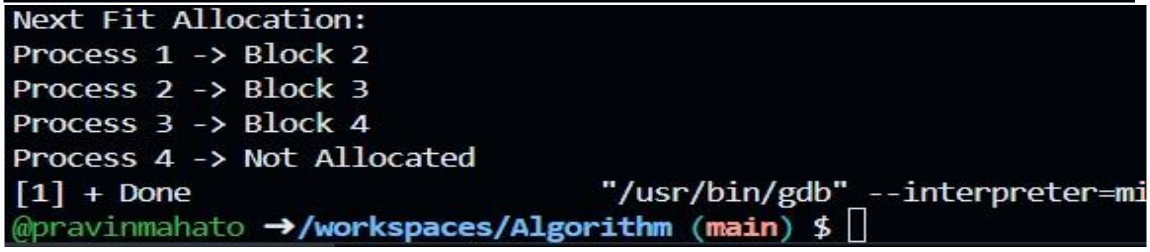
Assignment:-6

```
#include <stdio.h>
#define MAX 25

void firstFit(int bs[], int m, int ps[], int n) {
int allocation[MAX];
   int i, j;

   for (i = 0; i < n; i++) allocation[i] = -1;

   for (i = 0; i < n; i++) {
for (j = 0; j < m; j++) {
if (bs[j] >= ps[i]) {
allocation[i] = j;
bs[j] -= ps[i];
break;
         }
      }
   }

   printf("\nFirst Fit Allocation:\n");     for (i = 0; i < n; i++) {         if
(allocation[i] != -1)             printf("Process %d -> Block %d\n", i +
1, allocation[i] + 1);         else             printf("Process %d -> Not
Allocated\n", i + 1);
   }
}

void bestFit(int bs[], int m, int ps[], int n) {
int allocation[MAX], i, j;

   for (i = 0; i < n; i++) allocation[i] = -1;

   for (i = 0; i < n; i++) {
int bestIdx = -1;         for (j
= 0; j < m; j++) {             if
(bs[j] >= ps[i]) {
               if (bestIdx == -1 || bs[j] < bs[bestIdx])
bestIdx = j;
         }
      }         if (bestIdx != -1) {
allocation[i] = bestIdx;
         bs[bestIdx] -= ps[i];
      }
   }

   printf("\nBest Fit Allocation:\n");
    for (i = 0; i < n; i++) {         if (allocation[i] != -1)
printf("Process %d -> Block %d\n", i + 1, allocation[i] + 1);
else              printf("Process %d -> Not Allocated\n", i + 1);
    }
}

void worstFit(int bs[], int m, int ps[], int n) {
int allocation[MAX], i, j;

    for (i = 0; i < n; i++) allocation[i] = -1;

    for (i = 0; i < n; i++) {         int worstIdx = -1;
for (j = 0; j < m; j++) {             if (bs[j] >= ps[i]) {
if (worstIdx == -1 || bs[j] > bs[worstIdx])
worstIdx = j;
            }
        }
        if (worstIdx != -1) {
allocation[i] = worstIdx;
bs[worstIdx] -= ps[i];
        }
    }

    printf("\nWorst Fit Allocation:\n");     for (i = 0; i < n; i++) {
if (allocation[i] != -1)             printf("Process %d -> Block %d\n", i
+ 1, allocation[i] + 1);         else             printf("Process %d -> Not
Allocated\n", i + 1);
    }
}

void nextFit(int bs[], int m, int ps[], int n) {
int allocation[MAX];     int i, j, last_alloc =
0;

    for (i = 0; i < n; i++) allocation[i] = -1;

    for (i = 0; i < n; i++) {
int count = 0;
        for (j = last_alloc; count < m; j = (j + 1) % m, count++) {
           if (bs[j] >= ps[i]) {
allocation[i] = j;               bs[j] -=
ps[i];                last_alloc = (j + 1)
% m;                  break;
            }
        }
    }

    printf("\nNext Fit Allocation:\n");     for (i = 0; i < n; i++) {
if (allocation[i] != -1)             printf("Process %d -> Block %d\n", i
+ 1, allocation[i] + 1);         else             printf("Process %d -> Not
Allocated\n", i + 1);
    }
}

int main() {     int bno, pno, i;     int bs[MAX],
ps[MAX];     int bs1[MAX], bs2[MAX], bs3[MAX],
bs4[MAX];

    printf("Enter number of Memory Blocks: ");
scanf("%d", &bno);     printf("Enter size of
each Block:\n");     for (i = 0; i < bno; i++) {
printf("Block %d: ", i + 1);         scanf("%d",
&bs[i]);
    }

    printf("Enter number of Processes: ");
scanf("%d", &pno);     printf("Enter size
of each Process:\n");     for (i = 0; i <
pno; i++) {         printf("Process %d: ", i
+ 1);         scanf("%d", &ps[i]);
    }

    // Copy block sizes to use independently in each method
for (i = 0; i < bno; i++) {         bs1[i] = bs2[i] = bs3[i] =
bs4[i] = bs[i];
    }

    firstFit(bs1, bno, ps, pno);
bestFit(bs2, bno, ps, pno);
worstFit(bs3, bno, ps, pno);
nextFit(bs4, bno, ps, pno);

    return 0;
}
```

OUTPUT:-



```
Enter number of Memory Blocks: 6
Enter size of each Block:
Block 1: 200
Block 2: 400
Block 3: 600
Block 4: 500
Block 5: 300
Block 6: 250
Enter number of Processes: 4
Enter size of each Process:
Process 1: 357
Process 2: 210
Process 3: 499
Process 4: 468

First Fit Allocation:
Process 1 -> Block 2
Process 2 -> Block 3
Process 3 -> Block 4
Process 4 -> Not Allocated

Best Fit Allocation:
Process 1 -> Block 2
Process 2 -> Block 6
Process 3 -> Block 4
Process 4 -> Block 3

Worst Fit Allocation:
Process 1 -> Block 3
Process 2 -> Block 4
Process 3 -> Not Allocated
Process 4 -> Not Allocated
```



```
Next Fit Allocation:
Process 1 -> Block 2
Process 2 -> Block 3
Process 3 -> Block 4
Process 4 -> Not Allocated
[1] + Done                    "/usr/bin/gdb" --interpreter=mi
@pravinmahato →/workspaces/Algorithm (main) $
```