Dynamic Partitioning

Best Case

Code

```
#include <stdio.h>

void f(int blockSize[], int blocks, int processSize[], int processes)
{

   int allocation[processes]; //stores block no of allocated process
   for(int i = 0; i < processes; i++){
      allocation[i] = -1;
   }

   // pick each process and find suitable blocks
   // according to its size ad assign to it
   for (int i=0; i < processes; i++)
   {

      int indexPlaced = -1;
      for (int j=0; j < blocks; j++)
      {
         if (blockSize[j] >= processSize[i])
         {
            // place it at the first block fit to accomodate process
            if (indexPlaced == -1)
               indexPlaced = j;

            // if any future block is better that is
            // any future block with smaller size encountered
            // that can accomodate the given process
            else if (blockSize[j] < blockSize[indexPlaced])
               indexPlaced = j;
         }
      }

      // If we were successfully able to find block for the process
      if (indexPlaced != -1)
      {
         // allocate this block j to process p[i]
         allocation[i] = indexPlaced;

         // Reduce available memory for the block
         blockSize[indexPlaced] -= processSize[i];
      }
   }
```

```
    printf("\nProcess No.\tProcess Size\t\tBlock no.\n");
    for (int i = 0; i < processes; i++)
    {
        printf("%d \t\t\t %d \t\t\t", i+1, processSize[i]);
        if (allocation[i] != -1)
            printf("%d\n",allocation[i] + 1);
        else
            printf("Not Allocated\n");
    }
}



int main()
{

    int blockSize[10] = {};
    int processSize[10] = {};


    int np,nb;


    printf("Enter the no of blocks: ");
    scanf("%d",&nb);
    for (int i=0;i<nb;i++)
    {printf("Enter the size of block %d: ",(i+1));
    scanf("%d",&blockSize[i]);
    }

    printf("Enter the no of processes: ");
    scanf("%d",&np);
    for (int i=0;i<np;i++)
    {printf("Enter the size of block %d: ",(i+1));
    scanf("%d",&processSize[i]);
    }




    f(blockSize, nb, processSize, np);

    return 0 ;

}
```

Output

```
hadoop@hostssh:~/Music/21bce1070$ g++ os.c
hadoop@hostssh:~/Music/21bce1070$ ./a.out
Enter the no of blocks: 4
Enter the size of block 1: 100
Enter the size of block 2: 200
Enter the size of block 3: 300
Enter the size of block 4: 200
Enter the no of processes: 4
Enter the size of block 1: 150
Enter the size of block 2: 250
Enter the size of block 3: 50
Enter the size of block 4: 400

Process No.     Process Size            Block no.
1                       150                     2
2                       250                     3
3                       50                      2
4                       400                     Not Allocated
hadoop@hostssh:~/Music/21bce1070$
```

```
hadoop@hostssh:~/Music/21bce1070$ g++ os.c
hadoop@hostssh:~/Music/21bce1070$ ./a.out
Enter the no of blocks: 6
Enter the size of block 1: 200
Enter the size of block 2: 400
Enter the size of block 3: 600
Enter the size of block 4: 500
Enter the size of block 5: 300
Enter the size of block 6: 250
Enter the no of processes: 4
Enter the size of block 1: 357
Enter the size of block 2: 210
Enter the size of block 3: 468
Enter the size of block 4: 491

Process No.     Process Size            Block no.
1                       357                     2
2                       210                     6
3                       468                     4
4                       491                     3
```

Worst Case

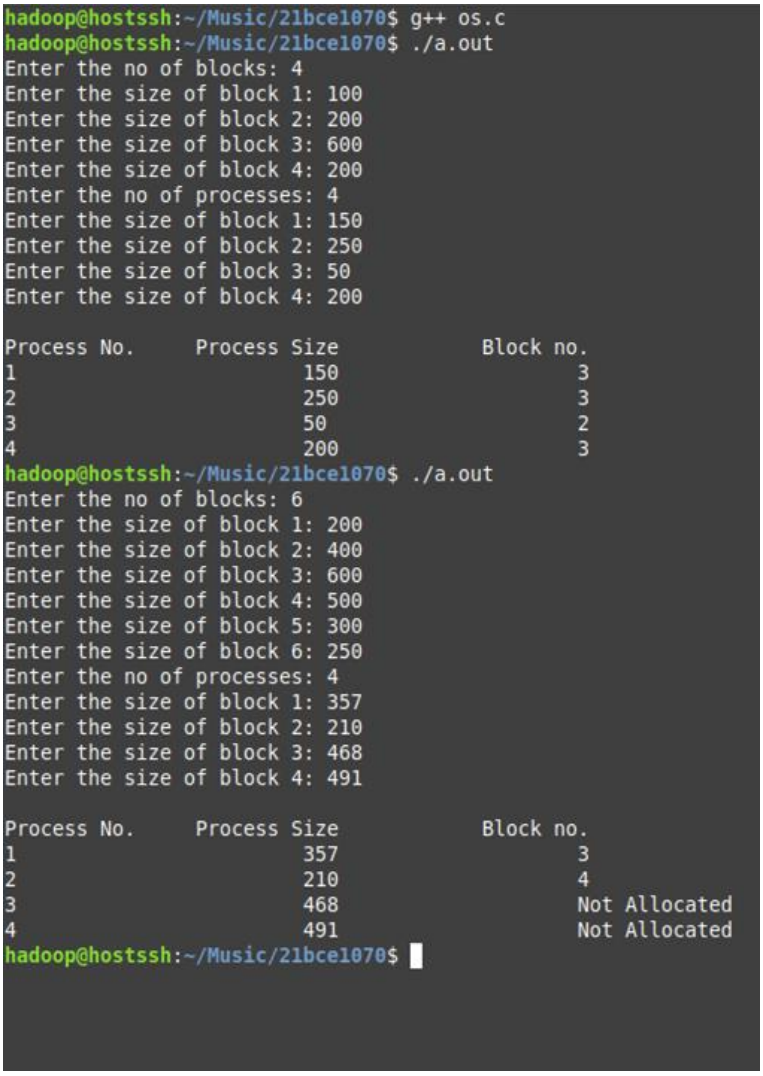
Code

```
#include <stdio.h>

void f(int blockSize[], int blocks, int processSize[], int processes)
{

    int allocation[processes]; //stores block no of allocated process
```

```
    for(int i = 0; i < processes; i++){
        allocation[i] = -1;
    }

    // pick each process and find suitable blocks
    // according to its size ad assign to it
    for (int i=0; i < processes; i++)
    {

        int indexPlaced = -1;
        for (int j=0; j < blocks; j++)
        {
            if (blockSize[j] >= processSize[i])
            {
                // place it at the first block fit to accomodate process
                if (indexPlaced == -1)
                    indexPlaced = j;

                // if any future block is better that is
                // any future block with smaller size encountered
                // that can accomodate the given process
                else if (blockSize[j] > blockSize[indexPlaced])
                    indexPlaced = j;
            }
        }

        // If we were successfully able to find block for the process
        if (indexPlaced != -1)
        {
            // allocate this block j to process p[i]
            allocation[i] = indexPlaced;

            // Reduce available memory for the block
            blockSize[indexPlaced] -= processSize[i];
        }
    }

    printf("\nProcess No.\tProcess Size\t\tBlock no.\n");
    for (int i = 0; i < processes; i++)
    {
        printf("%d \t\t\t %d \t\t\t", i+1, processSize[i]);
        if (allocation[i] != -1)
            printf("%d\n",allocation[i] + 1);
        else
            printf("Not Allocated\n");
    }
}



int main()
{
```

```
    int blockSize[10] = {};
    int processSize[10] = {};


    int np,nb;


    printf("Enter the no of blocks: ");
    scanf("%d",&nb);
    for (int i=0;i<nb;i++)
    {printf("Enter the size of block %d: ",(i+1));
    scanf("%d",&blockSize[i]);
    }

    printf("Enter the no of processes: ");
    scanf("%d",&np);
    for (int i=0;i<np;i++)
    {printf("Enter the size of block %d: ",(i+1));
    scanf("%d",&processSize[i]);
    }




    f(blockSize, nb, processSize, np);

    return 0 ;
}
```

Output



First Fit

Code

```
#include <stdio.h>

void f(int blockSize[], int blocks, int processSize[], int processes)
{

   int allocation[processes]; //stores block no of allocated process
   for(int i = 0; i < processes; i++){
      allocation[i] = -1;
   }
```

```
    // pick each process and find suitable blocks
    // according to its size ad assign to it
    for (int i=0; i < processes; i++)
    {

        int indexPlaced = -1;
        for (int j=0; j < blocks; j++)
        {
            if (blockSize[j] >= processSize[i])
            {
                // place it at the first block fit to accomodate process
                if (indexPlaced == -1)
                    indexPlaced = j;


            }
        }

        // If we were successfully able to find block for the process
        if (indexPlaced != -1)
        {
            // allocate this block j to process p[i]
            allocation[i] = indexPlaced;

            // Reduce available memory for the block
            blockSize[indexPlaced] -= processSize[i];
        }
    }

    printf("\nProcess No.\tProcess Size\t\tBlock no.\n");
    for (int i = 0; i < processes; i++)
    {
        printf("%d \t\t\t %d \t\t\t", i+1, processSize[i]);
        if (allocation[i] != -1)
            printf("%d\n",allocation[i] + 1);
        else
            printf("Not Allocated\n");
    }
}



int main()
{

    int blockSize[10] = {};
    int processSize[10] = {};


    int np,nb;
```

```
    printf("Enter the no of blocks: ");
    scanf("%d",&nb);
    for (int i=0;i<nb;i++)
    {printf("Enter the size of block %d: ",(i+1));
    scanf("%d",&blockSize[i]);
    }

    printf("Enter the no of processes: ");
    scanf("%d",&np);
    for (int i=0;i<np;i++)
    {printf("Enter the size of block %d: ",(i+1));
    scanf("%d",&processSize[i]);
    }




    f(blockSize, nb, processSize, np);

    return 0 ;
}
```

Output

```
hadoop@hostssh:~/Music/21bce1070$ g++ os.c
hadoop@hostssh:~/Music/21bce1070$ ./a.out
Enter the no of blocks: 4
Enter the size of block 1: 200
Enter the size of block 2: 150
Enter the size of block 3: 100
Enter the size of block 4: 200
Enter the no of processes: 4
Enter the size of block 1: 100
Enter the size of block 2: 75
Enter the size of block 3: 125
Enter the size of block 4: 25

Process No.     Process Size            Block no.
1                       100                     1
2                       75                      1
3                       125                     2
4                       25                      1
hadoop@hostssh:~/Music/21bce1070$ ./a.out
Enter the no of blocks: 6
Enter the size of block 1: 200
Enter the size of block 2: 400
Enter the size of block 3: 600
Enter the size of block 4: 500
Enter the size of block 5: 300
Enter the size of block 6: 250
Enter the no of processes: 4
Enter the size of block 1: 357
Enter the size of block 2: 210
Enter the size of block 3: 468
Enter the size of block 4: 491

Process No.     Process Size            Block no.
1                       357                     2
2                       210                     3
3                       468                     4
4                       491                     Not Allocated
hadoop@hostssh:~/Music/21bce1070$
```