Program for First Fit algorithm in Memory Management

```
#include<bits/stdc++.h>
using namespace std;

void firstFit(int blockSize[], int m,
              int processSize[], int n)
{
   int allocation[n];
   memset(allocation, -1, sizeof(allocation));
   for (int i = 0; i < n; i++)
   {
      for (int j = 0; j < m; j++)
      {
         if (blockSize[j] >= processSize[i])
         {
            allocation[i] = j;
            blockSize[j] -= processSize[i];
            break;
         }
      }
   }
   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 blockSize[] = {100, 500, 200, 300, 600};
   int processSize[] = {212, 417, 112, 426};
   int m = sizeof(blockSize) / sizeof(blockSize[0]);
   int n = sizeof(processSize) / sizeof(processSize[0]);

   firstFit(blockSize, m, processSize, n);

   return 0 ;
```

```
}
```

OUTPUT :-

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

Program for Next Fit algorithm in Memory Management

```
#include <bits/stdc++.h>
using namespace std;

void NextFit(int blockSize[], int m, int processSize[], int n)
{
   int allocation[n], j = 0;

   memset(allocation, -1, sizeof(allocation));
    for (int i = 0; i < n; i++) {

      while (j < m) {
         if (blockSize[j] >= processSize[i]) {
            allocation[i] = j;
            blockSize[j] -= processSize[i];
            break;
         }

         j = (j + 1) % m;
      }
   }
   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 blockSize[] = { 5, 10, 20 };
    int processSize[] = { 10, 20, 5 };
    int m = sizeof(blockSize) / sizeof(blockSize[0]);
    int n = sizeof(processSize) / sizeof(processSize[0]);

    NextFit(blockSize, m, processSize, n);

    return 0;
}
```

OUTPUT :-

Process No.	Process Size	Block no.
1	10	2
2	20	3
3	5	1

Program for Best Fit algorithm in Memory Management

```
#include<bits/stdc++.h>
using namespace std;

void bestFit(int blockSize[], int m, int processSize[], int n)
{
    int allocation[n];

    memset(allocation, -1, sizeof(allocation));
    for (int i=0; i<n; i++)
    {
        int bestIdx = -1;
        for (int j=0; j<m; j++)
        {
            if (blockSize[j] >= processSize[i])
            {
                if (bestIdx == -1)
                    bestIdx = j;
                else if (blockSize[bestIdx] > blockSize[j])
                    bestIdx = j;
            }
        }
        if (bestIdx != -1)
        {
            allocation[i] = bestIdx;
            blockSize[bestIdx] -= 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 blockSize[] = {100, 500, 200, 300, 600};
    int processSize[] = {212, 417, 112, 426};
    int m = sizeof(blockSize)/sizeof(blockSize[0]);
    int n = sizeof(processSize)/sizeof(processSize[0]);

    bestFit(blockSize, m, processSize, n);

    return 0 ;
}
```

OUTPUT :-

Process No.	Process Size	Block no.
1	212	4
2	417	2
3	112	3
4	426	5

Program for Worst Fit algorithm in Memory Management

```
#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 blockSize[] = {100, 500, 200, 300, 600};
        int processSize[] = {212, 417, 112, 426};
        int m = sizeof(blockSize)/sizeof(blockSize[0]);
        int n = sizeof(processSize)/sizeof(processSize[0]);

        worstFit(blockSize, m, processSize, n);

        return 0 ;
}
```

OUTPUT :-

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