BEST FIT

Aim:

To implement Best Fit memory allocation technique using Python.

Algorithm:

1. Input memory blocks and processes with sizes
2. Initialize all memory blocks as free.
3. Start by picking each process and find the minimum block size that can be assigned to
current process
4. If found then assign it to the current process.
5. If not found then leave that process and keep checking the further processes.

Program Code:

```
#include<stdio.h>
#include<string.h>
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];
      }
   }
```

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