# **Lab 9 – Scheduling Algorithms**

**First Fit Algorithm**

#include<stdio.h>

void main()

{

int bsize[10], psize[10], bno, pno, flags[10], allocation[10], i, j;

for(i = 0; i < 10; i++)

{

flags[i] = 0;

allocation[i] = -1;

}

printf("Enter no. of blocks: ");

scanf("%d", &bno);

printf("\nEnter size of each block: ");

for(i = 0; i < bno; i++)

scanf("%d", &bsize[i]);

printf("\nEnter no. of processes: ");

scanf("%d", &pno);

printf("\nEnter size of each process: ");

for(i = 0; i < pno; i++)

scanf("%d", &psize[i]);

for(i = 0; i < pno; i++) //allocation as per first fit

for(j = 0; j < bno; j++)

if(flags[j] == 0 && bsize[j] >= psize[i])

{

allocation[j] = i;

flags[j] = 1;

break;

}

//display allocation details

printf("\nBlock no.\tsize\t\tprocess no.\t\tsize");

for(i = 0; i < bno; i++)

{

printf("\n%d\t\t%d\t\t", i+1, bsize[i]);

if(flags[i] == 1)

printf("%d\t\t\t%d",allocation[i]+1,psize[allocation[i]]);

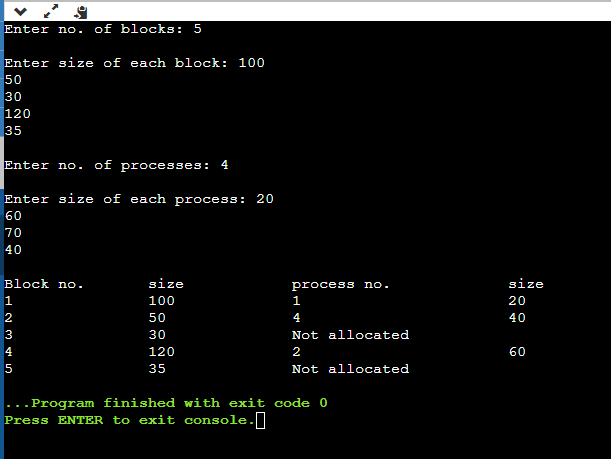
else

printf("Not allocated");

}

}

**Output**



**Best Fit Algorithm:**

#include<stdio.h>

void main()

{

int fragment[20],b[20],p[20],i,j,nb,np,temp,lowest=9999;

static int barray[20],parray[20];

printf("\n\t\t\tMemory Management Scheme - Best Fit");

printf("\nEnter the number of blocks:");

scanf("%d",&nb);

printf("Enter the number of processes:");

scanf("%d",&np);

printf("\nEnter the size of the blocks:-\n");

for(i=1;i<=nb;i++)

{

printf("Block no.%d:",i);

scanf("%d",&b[i]);

}

printf("\nEnter the size of the processes :-\n");

for(i=1;i<=np;i++)

{

printf("Process no.%d:",i);

scanf("%d",&p[i]);

}

for(i=1;i<=np;i++)

{

for(j=1;j<=nb;j++)

{

if(barray[j]!=1)

{

temp=b[j]-p[i];

if(temp>=0)

if(lowest>temp)

{

parray[i]=j;

lowest=temp;

}

}

}

fragment[i]=lowest;

barray[parray[i]]=1;

lowest=10000;

}

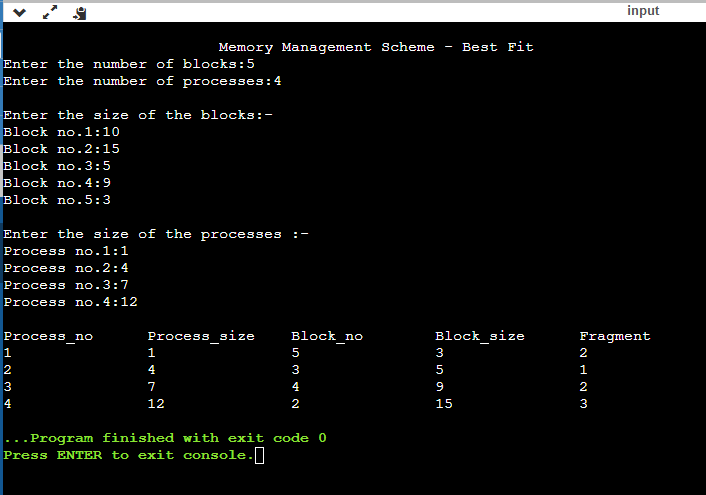
printf("\nProcess\_no\tProcess\_size\tBlock\_no\tBlock\_size\tFragment");

for(i=1;i<=np && parray[i]!=0;i++)

printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d",i,p[i],parray[i],b[parray[i]],fragment[i]);

}

**Output:**



**Worst Fit Algorithm:**

#include<stdio.h>

int main()

{

int fragments[10], blocks[10], files[10];

int m, n, number\_of\_blocks, number\_of\_files, temp, top = 0;

static int block\_arr[10], file\_arr[10];

printf("\nEnter the Total Number of Blocks:\t");

scanf("%d",&number\_of\_blocks);

printf("Enter the Total Number of Files:\t");

scanf("%d",&number\_of\_files); printf("\nEnter the Size of the Blocks:\n");

for(m = 0; m < number\_of\_blocks; m++)

{

printf("Block No.[%d]:\t", m + 1);

scanf("%d", &blocks[m]);

}

printf("Enter the Size of the Files:\n"); for(m = 0; m < number\_of\_files; m++)

{

printf("File No.[%d]:\t", m + 1);

scanf("%d", &files[m]);

}

for(m = 0; m < number\_of\_files; m++){ for(n = 0; n < number\_of\_blocks; n++)

{

if(block\_arr[n] != 1)

{

temp = blocks[n] - files[m]; if(temp >= 0)

{

if(top < temp)

{

file\_arr[m] = n; top = temp;

}

}

}

fragments[m] = top; block\_arr[file\_arr[m]] = 1;

top = 0;

}

}

printf("\nFile Number\tFile Size\tBlock Number\tBlock Size\tFragment"); for(m = 0; m < number\_of\_files; m++)

{

printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d", m, files[m], file\_arr[m], blocks[file\_arr[m]], fragments[m]);

}

printf("\n"); return 0;}

**Output:**

