**Name : Saqib Shehzad**

**Registration No : 2021-CS-187**

**Week 10 Lab Task:**

**Solution :**

**Code: 1**

#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;

}

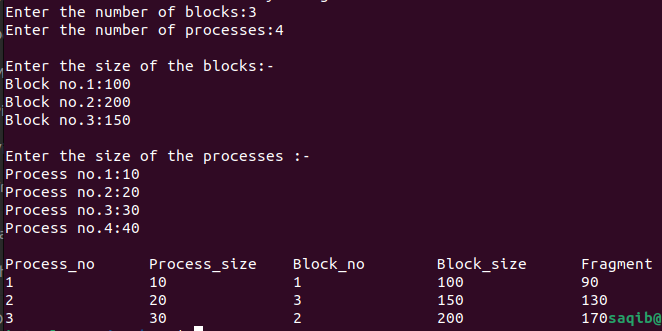
printf("\nProcess\_no\tProcess\_size\tBlock\_no\tBlock\_size\tFragment");

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 :**

****

**Code:2**

#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]]);

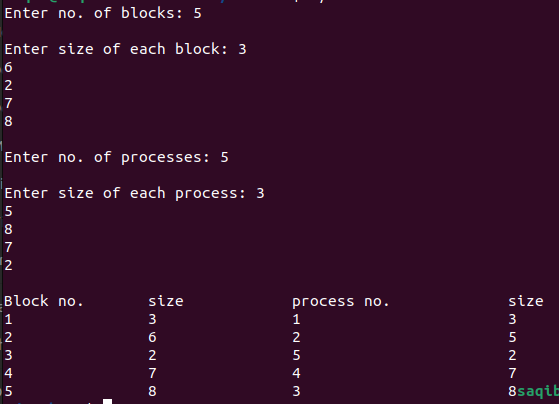
else

printf("Not allocated");

}

}

Output:



Code:3

#include <stdio.h>

void implimentWorstFit(int blockSize[], int blocks, int processSize[], int processes)

{

// This will store the block id of the allocated block to a process

int allocation[processes];

int occupied[blocks];

// initially assigning -1 to all allocation indexes

// means nothing is allocated currently

for(int i = 0; i < processes; i++){

allocation[i] = -1;

}

for(int i = 0; i < blocks; i++){

occupied[i] = 0;

}

// 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 not occupied and block size is large enough

if(blockSize[j] >= processSize[i] && !occupied[j])

{

// place it at the first block fit to accomodate process

if (indexPlaced == -1)

indexPlaced = j;

// if any future block is larger than the current block where

// process is placed, change the block and thus indexPlaced

else if (blockSize[indexPlaced] < blockSize[j])

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;

// make the status of the block as occupied

occupied[indexPlaced] = 1;

// Reduce available memory for the block

blockSize[indexPlaced] -= processSize[i];

}

}

printf("\nProcess No.\tProcess Size\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");

}

}

// Driver code

int main()

{

int blockSize[] = {100, 50, 30, 120, 35};

int processSize[] = {40, 10, 30, 60};

int blocks = sizeof(blockSize)/sizeof(blockSize[0]);

int processes = sizeof(processSize)/sizeof(processSize[0]);

implimentWorstFit(blockSize, blocks, processSize, processes);

return 0;

}

Output:

