WEEK 7 OS LAB

Write a C program to simulate the following contiguous memory

allocation techniques

a) Worst-fit

b) Best-fit

c) First-fit

source code:-

#include <stdio.h>

#include<stdlib.h>

#define max 25

void readInput(int \*nb, int \*nf, int b[], int f[]);

void bestFit(int nb, int nf, int b[], int f[], int bf[], int ff[], int frag[]);

void worstFit(int nb, int nf, int b[], int f[], int bf[], int ff[], int frag[]);

void firstFit(int nb, int nf, int b[], int f[], int bf[], int ff[], int frag[]);

void displayResults(int nf, int f[], int b[], int ff[]);

int main()

{

int nb, nf, ch;

int b[max], f[max], bf[max] = {0}, ff[max] = {0}, frag[max] = {0};

readInput(&nb, &nf, b, f);

printf("1.Best Fit 2.Worst Fit 3.First Fit 4. Exit\n");

scanf("%d",&ch);

switch(ch)

{

case 1: bestFit(nb, nf, b, f, bf, ff, frag);

break;

case 2: worstFit(nb, nf, b, f, bf, ff, frag);

break;

case 3: firstFit(nb, nf, b, f, bf, ff, frag);

break;

case 4: exit(0);

break;

default: printf("Inavlid choice\n");

break;

}

displayResults(nf, f, b, ff);

return 0;

}

void readInput(int \*nb, int \*nf, int b[], int f[])

{

int i;

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

scanf("%d", nb);

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

scanf("%d", nf);

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

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

{

printf("Block %d:", i);

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

}

printf("Enter the size of the files:\n");

for (i = 1; i <= \*nf; i++)

{

printf("File %d:", i);

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

}

}

void bestFit(int nb, int nf, int b[], int f[], int bf[], int ff[], int frag[])

{

int i, j, temp, lowest = 10000;

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

{

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

{

if (bf[j] != 1) //if bf[j] is not allocated

{

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

if (temp >= 0)

{

if(lowest > temp)

{

ff[i] = j;

lowest = temp;

}

}

}

}

frag[i] = lowest;

bf[ff[i]] = 1;

lowest = 10000;

}

}

void worstFit(int nb, int nf, int b[], int f[], int bf[], int ff[], int frag[])

{

int i, j, temp, lowest = 10000;

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

{

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

{

if (bf[j] != 1)

{

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

if (temp >= 0)

{

if (lowest == 10000 || temp > lowest)

{

ff[i] = j;

lowest = temp;

}

}

}

}

frag[i] = lowest;

bf[ff[i]] = 1;

lowest = 10000;

}

}

void firstFit(int nb, int nf, int b[], int f[], int bf[], int ff[], int frag[])

{

int i, j, temp;

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

{

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

{

if (bf[j] != 1)

{

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

if (temp >= 0)

{

ff[i] = j;

break;

}

}

}

frag[i] = temp;

bf[ff[i]] = 1;

}

}

void displayResults(int nf, int f[], int b[], int ff[])

{

int i;

printf("\nFile\_no\t\tFile\_size\tBlock\_size");

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

{

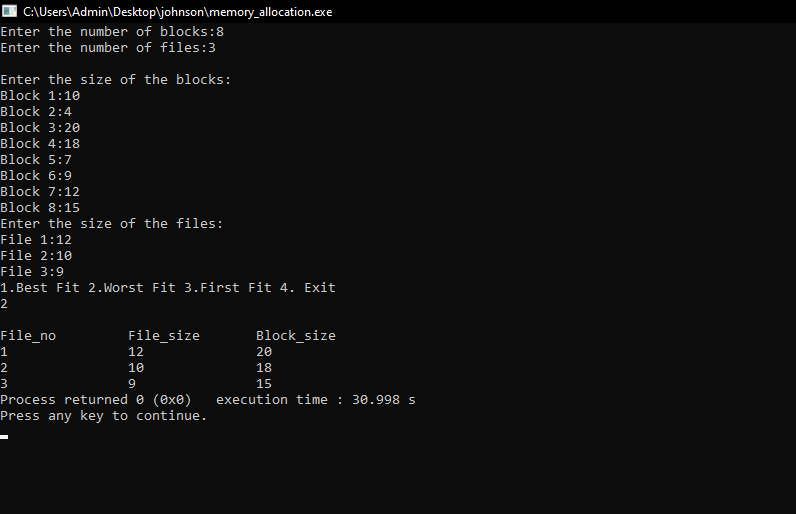
printf("\n%d\t\t%d\t\t%d", i, f[i], b[ff[i]]);

}

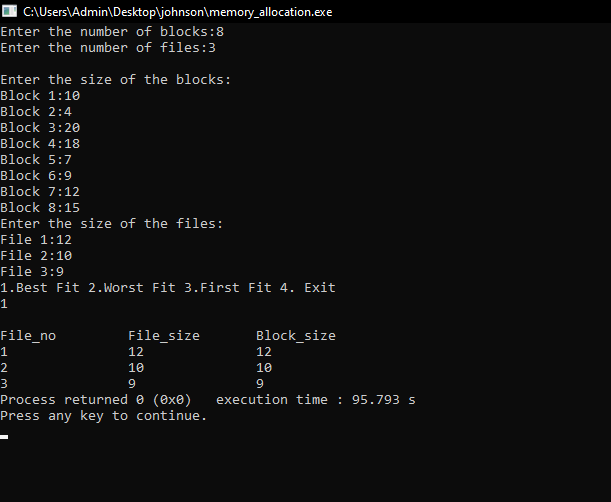
}

OUTPUT:-

WORST FIT



BEST FIT



FIRST FIT

