WEEK9&10

WEEK 9

Write a C program to simulate the following contiguous memory allocation techniques:

(a) Worst-fit

(b) Best-fit

(c) First-fit

1.1.1 Code:

```
#include <stdio.h>


#define max 25


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

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

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


int main()

{

	int b[max], f[max], nb, nf;


	printf("Memory Management Schemes\n");
```

```
printf("\nEnter 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 (int i = 1; i <= nb; i++)
{
printf("Block %d:", i);
scanf("%d", &b[i]);
}

printf("\nEnter the size of the files:\n");
for (int i = 1; i <= nf; i++)
{
printf("File %d:", i);
scanf("%d", &f[i]);
}
```

```
	printf("\nMemory Management Scheme - First Fit");
	firstFit(b, nb, f, nf);

	printf("\n\nMemory Management Scheme - Worst Fit");
	worstFit(b, nb, f, nf);

	printf("\n\nMemory Management Scheme - Best Fit");
	bestFit(b, nb, f, nf);

	return 0;
}

void firstFit(int b[], int nb, int f[], int nf)
{
	int bf[max] = {0};
	int ff[max] = {0};
	int frag[max], i, j;

	for (i = 1; i <= nf; i++)
	{
	for (j = 1; j <= nb; j++)
```

```
        {
        if (bf[j] != 1 && b[j] >= f[i])
        {
                ff[i] = j;
                bf[j] = 1;
                frag[i] = b[j] - f[i];
                break;
        }
        }
        }

    printf("\nFile_no:\tFile_size:\tBlock_no:\tBlock_size:\tFragment");
        for (i = 1; i <= nf; i++)
        {
       printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d", i, f[i], ff[i], b[ff[i]], frag[i]);
        }
}

void worstFit(int b[], int nb, int f[], int nf)
{
        int bf[max] = {0};
```

```
int ff[max] = {0};
int frag[max], i, j, temp, highest = 0;

for (i = 1; i <= nf; i++)
{
for (j = 1; j <= nb; j++)
{
if (bf[j] != 1)
{
        temp = b[j] - f[i];
        if (temp >= 0 && highest < temp)
        {
        ff[i] = j;
        highest = temp;
        }
}
}
frag[i] = highest;
bf[ff[i]] = 1;
highest = 0;
}
```

```
    printf("\nFile_no:\tFile_size:\tBlock_no:\tBlock_size:\tFragment");
        for (i = 1; i <= nf; i++)
        {
       printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d", i, f[i], ff[i], b[ff[i]], frag[i]);
        }
}

void bestFit(int b[], int nb, int f[], int nf)
{
        int bf[max] = {0};
        int ff[max] = {0};
        int frag[max], 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 && lowest > temp)
            {
            ff[i] = j;
            lowest = temp;
            }
        }
        }
        frag[i] = lowest;
        bf[ff[i]] = 1;
        lowest = 10000;
        }

        printf("\nFile_no:\tFile_size:\tBlock_no:\tBlock_size:\tFragment");
        for (i = 1; i <= nf && ff[i] != 0; i++)
        {
      printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d", i, f[i], ff[i], b[ff[i]], frag[i]);
        }
}
```

OUTPUT

```
Memory Management Schemes

Enter the number of blocks:3
Enter the number of files:2

Enter the size of the blocks:
Block 1:5
Block 2:2
Block 3:7

Enter the size of the files:
File 1:1
File 2:4

Memory Management Scheme - First Fit
File_no:        File_size:      Block_no:       Block_size:     Fragment
1               1               1               5               4
2               4               3               7               3

Memory Management Scheme - Worst Fit
File_no:        File_size:      Block_no:       Block_size:     Fragment
1               1               3               7               6
2               4               1               5               1

Memory Management Scheme - Best Fit
File_no:        File_size:      Block_no:       Block_size:     Fragment
1               1               2               2               1
2               4               1               5               1
```

WEEK10

Write a C program to simulate paging technique of memory management.

1.1.1 Code:

```
#include<stdio.h>
#define MAX 50
int main()
{
        int page[MAX],i,n,f,ps,off,pno;
        int choice=0;
        printf("Enter the number of pages in memory: ");
        scanf("%d",&n);
        printf("\nEnter Page size: ");
        scanf("%d",&ps);

        printf("\nEnter number of frames: ");
        scanf("%d",&f);
        for(i=0;i<n;i++)
        page[i]=-1;
```

```
printf("\nEnter the Page Table\n");
printf("(Enter frame no as -1 if that page is not present in any frame)\n\n");

printf("\nPage No\t\tFrame No\n-------\t\t-------");
for(i=0;i<n;i++)
{
printf("\n\n%d\t\t",i);
scanf("%d",&page[i]);
}

do
{
printf("\n\nEnter the logical address(i.e,page no & offset):");
scanf("%d%d",&pno,&off);

if(page[pno]==-1)
printf("\n\nThe required page is not available in any of frames");
else
printf("\nPhysical address(i.e,frame no & offset):%d,%d",page[pno],off);

printf("\n\nDo you want to continue(1/0)?:");
```

```
        scanf("%d",&choice);
        }while(choice==1);

        return 1;
}
```

```
Enter the number of pages in memory: 4

Enter Page size: 10

Enter number of frames: 4

Enter the Page Table
(Enter frame no as -1 if that page is not present in any frame)


Page No         Frame No
-------         -------

0               -1

1               8

2               5

3               2
```

a) write a program to simulate following contiguous memory allocation techniques

a) worst - fit
b) best fit
c) first fit

worst fit

```
#include <stdio.h>
void worstfit (int block size[], int blocks, int process[],
    int processes);
{
  int allocation [processes];
  int occupied [blocks];
  for (int i=0; i< process; i++){
      allocation [i] = -1;
  }
  for (int i=0; i < blocks; i++) {
      occupied [i] = 0;
  }
  for (int i=0; i < processes; i++) {
    int index proces = -1;
    for (int j=0; j < blocks size [i] && 10 occupied [j]
{
  if (index placed == -1)
      index placed = j;
else if (block size [index] = process size [i] && 0
      occupied [j])
    if (new index place = j;
}
}
```

```
if (index_Need != -1) {
    allocation[i] = index_Need;
    occupied[index_Need] = 1
    block_Size[index_Need] = process_Size[i];
  }
}
printf("\nProcess No \t Process Size \t Block no: \n");
for (int i = 0; i < process; (++f) {
  printf("%d \t \t %d \t \t \t " i+1, Process_Size[i]);
  if (allocation[i] != -1)
  printf("%d \n", allocation[i] + 1);
else
  printf("Not allocated \n");
}
}
int main() {
  int i, blocks, processes;
printf("Enter no of blocks: ");
Scanf("%d", &blocks);
int block_Size[blocks];
printf("\n Enter Size of each block: ");
for (int i = 0; i < blocks; i++)
Scanf("%d", &process);
  int process_size[process];
  printf("\n Enter Size of each process: ");
for (i = 0; i < process; i++)
Scanf("%d", process_Size[i]);
worstfit(block_size, blocks, process_size, processes);
  return 0;
}
```

Output

enter no of blocks : 3

enter size of each block : 5 2 7

enter no of processes : 2

enter size of each process : 1 4

process no	Process Size	Block no
1	1	3
2	4	1

b) #include <stdio.h>

#define MAX 16

void Bestfit (int blockSize[], int blocks, int processSize[], int processes, int m) {

int allocation[processes];

int occupied[blocks];

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

allocation[i] = i;

}

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

occupied[i] = 0;

}

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

int indexPlaced = -1;

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

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

if (indexPlaced == -1)

indexPlaced = j;

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

indexPlaced = j;

}

```
    if (index placed != -1)
    {
      allocation[i] = indexplaced;
      occupied[indexplaced] = 1;
    }
  }
  printf("\n Process No.\t Process Size\t Block no\n");
  for(int i = 0; i < process; i++){
    printf("%d\t\t%d\t\t", i+1, process size[i]);
    if (allocation[i] != -1)
    printf("%d\n", allocation[i] + 1);
    else
      printf("Not allocated\n");
  }
}
int main()
{
  int p, m, j;
  printf("Enter no of process & blocks: ");
  scanf("%d %d", &p, &m);
  printf("Enter process sizes: ");
  for(j = 0; j < p; j++)
  printf("Enter the block sizes: ");
  for(j = 0; j < m; j++)
  scanf("%d", &block size[j]);
  int blocks = size of(block size)/size of(block size[0]);
int process = size of(process size)/size of(process size[0]);
Best fit(block size, blocks, process size, process, m);
  return 0;
}
```

Output :-

enter the no of process & blocks : 2 3

enter the process size : 1 4

enter the block sizes : 5 2 3

Process no	process size	Block no
1		2
2	4	1

```
#include <stdio.h>
#include <conio.h>
#define max 25
void main() {
int frag[max], b[max], f[max], i, j, nb, nf
Static int bf[max], ff[max];
printf("\n\t Memory management scheme - First Fit");
printf("Enter the no of blocks");
Scanf("%d", &nb);
printf("Enter no of files: ");
Scanf("%d", &nf);
printf("\n Enter the size of blocks: \n");
for(i=1; i<=nb; i++){
    printf("Block %d:", i);
    Scanf("%d", &b[i]);
}
printf("Enter the size of files: \n");
for(i=1; i<=nf; i++){
printf("File %d:", i);
    Scanf("%d", &f[i]);
```

```
for (i=1; i<=nb; i++) {
  if (bt[j] != 1) {
    if (temp >= 0) {
      ff[i] = j;
      flag[i] = b[j] - f[i];
      allocated = 1
      printf("\n%d\t\t%d\t\t%d\t\t%d\t\t",
      i, f[i], f[i], b[ff[i]]);
      break;
    }
  }
}
  if (!allocated) {
  printf("\n %d \t\t %d \t\t Not Allocated \t\t
}
}
getch();
}
```

output

Memory Management Scheme - First fit

Enter the no of blocks : 3

enter the no of files : 2

enter the size of the blocks ;

Block 1 : 5

Block 2 : 2

Block 3 : 2

enter the size of files

File 1 : 1

File 2 : 4

File no	file-size	Block no	Block size
1	1	1	5
2	4	3	2

10/10

23/8/23