NAME: SPANDAN MUKHERJEE

REGISTRATION NUMBER: 21BCE1132

SUBJECT: OPERATING SYSTEM LAB

TOPIC: OS-MEMORY-MANAGEMENT

Develop a C program to do best fit, worst fit, first fit memory allocation of fixed partition.

a)Best Fit

CODE:
```
#include<stdio.h>
#include<conio.h>
#define max 25
int main()
{
int frag[max],b[max],f[max],i,j,nb,nf,temp,lowest=10000;
static int bf[max],ff[max];
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(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]);
}
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>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]);
return 0;
}
```

OUTPUT:

```
spandan@spandan-VirtualBox: ~
spandan@spandan-VirtualBox:~$ gedit bestfit.c
^C
spandan@spandan-VirtualBox:~$ gcc bestfit.c
spandan@spandan-VirtualBox:~$ ./a.out

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

Enter the size of the blocks:-
Block 1:2
Block 2:1
Block 3:2
Enter the size of the files :-
File 1:2
File 2:3

File No File Size       Block No        Block Size      Fragment
1               2               1               2               0spandan@spandan-VirtualBox:~$ ./a.out

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

Enter the size of the blocks:-
Block 1:2
Block 2:3
Enter the size of the files :-
File 1:1
File 2:2

File No File Size       Block No        Block Size      Fragment
1               1               1               2               1
spandan@spandan-VirtualBox:~$
```

b) FIRST-FIT

CODE:

```
#include<stdio.h>

#define max 25
int main()
{
int frag[max],b[max],f[max],i,j,nb,nf,temp;
static int bf[max],ff[max];
printf("\n\tMemory Management Scheme - First Fit");
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(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]);
}
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;
}
printf("\nFile_no:\tFile_size :\tBlock_no:\tBlock_size:\tFragement");
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]);
return 0;
}
```

OUTPUT:

```
spandan@spandan-VirtualBox: ~
spandan@spandan-VirtualBox:~$ gedit firstfit.c
^C
spandan@spandan-VirtualBox:~$ gcc firstfit.c
spandan@spandan-VirtualBox:~$ ./a.out

        Memory Management Scheme - First Fit
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

File_no:        File_size :     Block_no:       Block_size:     Fragement
1               1               1               5               4
2               4               3               7               3spandan@spandan
spandan@spandan-VirtualBox:~$
```

c) WORST FIT

CODE:

```
#include<stdio.h>

#define max 25

int main()

{

int frag[max],b[max],f[max],i,j,nb,nf,temp,highest=0;

static int bf[max],ff[max];

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

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");
```

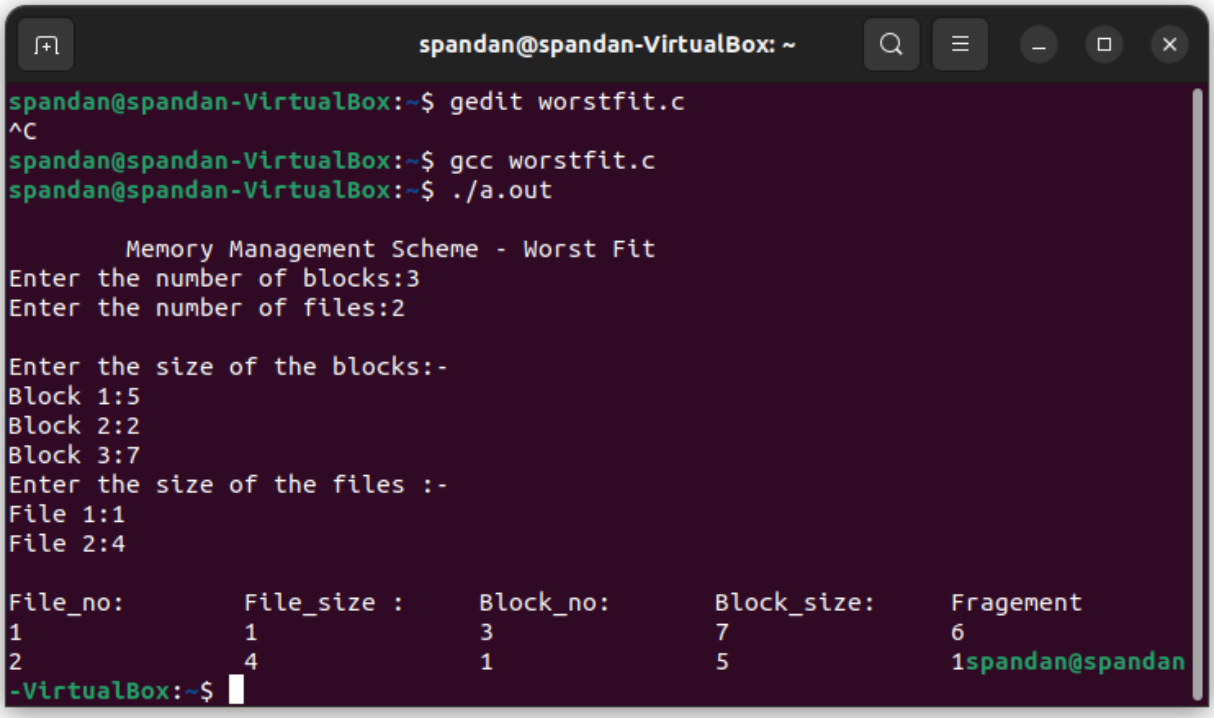
```
for(i=1;i<=nb;i++)
{
printf("Block %d:",i);
scanf("%d",&b[i]);
}
printf("Enter the size of the files :-\n");
for(i=1;i<=nf;i++)
{
printf("File %d:",i);
scanf("%d",&f[i]);
}
for(i=1;i<=nf;i++)
{
for(j=1;j<=nb;j++)
{
if(bf[j]!=1) //if bf[j] is not allocated
{
temp=b[j]-f[i];
if(temp>=0)
if(highest<temp)
{
ff[i]=j;
highest=temp;
}
}
}
frag[i]=highest;
bf[ff[i]]=1;
highest=0;
}
printf("\nFile_no:\tFile_size :\tBlock_no:\tBlock_size:\tFragement");
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]);

return 0;

}
```

OUTFIT:




Develop a C program to do best fit, worst fit, first fit memory allocation of fixed partition using 3 threads and and threadys may return the name of the processes which are unallocated if any.

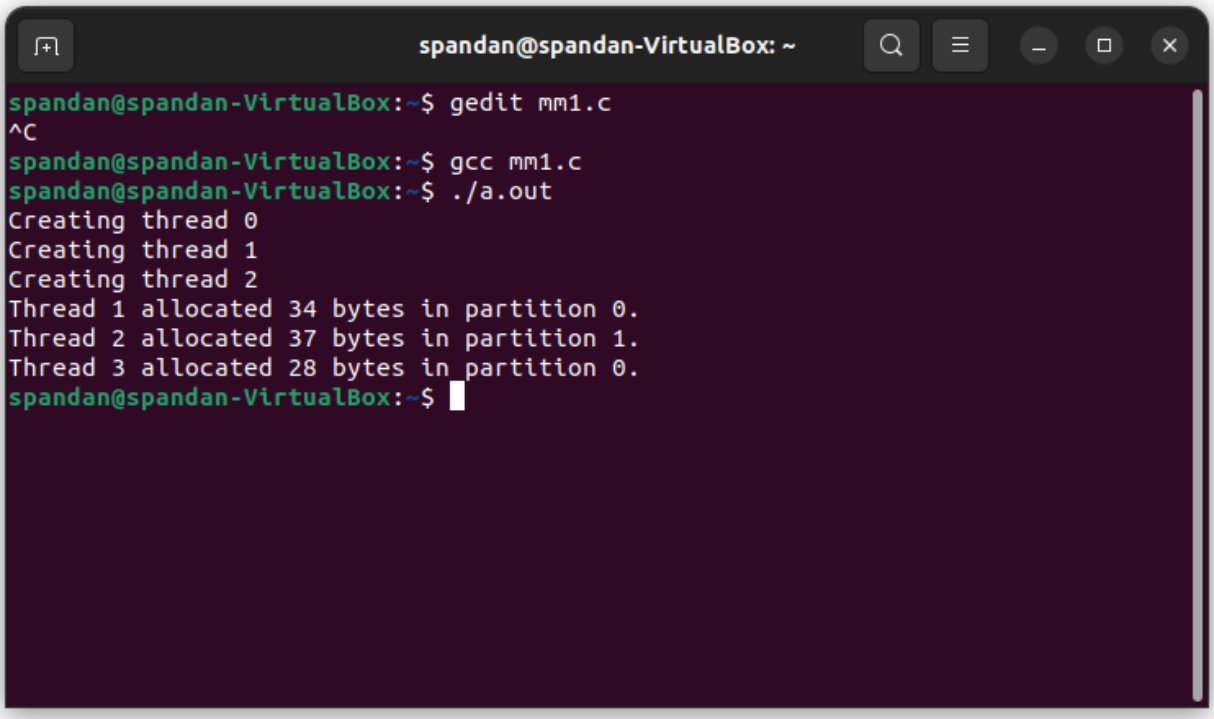
CODE:

```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#define NUM_PARTITIONS 10
#define PARTITION_SIZE 100
#define NUM_THREADS 3
int memory[NUM_PARTITIONS][PARTITION_SIZE];
int free_list[NUM_PARTITIONS];
pthread_mutex_t mutex;
void initialize_memory() {
for (int i = 0; i < NUM_PARTITIONS; i++) {
free_list[i] = 1;
}
}
int best_fit(int size) {
int best_fit_index = -1;
```

```
int best_fit_size = PARTITION_SIZE + 1;
for (int i = 0; i < NUM_PARTITIONS; i++) {
if (free_list[i] && (PARTITION_SIZE - size) < best_fit_size) {
best_fit_index = i;
best_fit_size = PARTITION_SIZE - size;
}
}
return best_fit_index;
}
int worst_fit(int size) {
int worst_fit_index = -1;
int worst_fit_size = -1;
for (int i = 0; i < NUM_PARTITIONS; i++) {
if (free_list[i] && (PARTITION_SIZE - size) > worst_fit_size) {
worst_fit_index = i;
worst_fit_size = PARTITION_SIZE - size;
}
}
return worst_fit_index;
}
int first_fit(int size) {
for (int i = 0; i < NUM_PARTITIONS; i++) {
if (free_list[i] && (PARTITION_SIZE - size) >= 0) {
return i;
}
}
return -1;
}
void *allocate_memory(void *threadid) {
int tid = *(int*)threadid;
int partition_index;
int size = (rand() % (PARTITION_SIZE/2)) + 1;
switch (tid) {
case 0:
partition_index = best_fit(size);
break;
case 1:
partition_index = worst_fit(size);
break;
case 2:
partition_index = first_fit(size);
break;
}
if (partition_index >= 0) {
pthread_mutex_lock(&mutex);
free_list[partition_index] = 0;
printf("Thread %d allocated %d bytes in partition %d.\n", tid, size, partition_index);
pthread_mutex_unlock(&mutex);
} else {
```

```
printf("Thread %d could not allocate %d bytes.\n", tid, size);
}
pthread_exit(NULL);
}
int main(int argc, char *argv[]) {
pthread_t threads[NUM_THREADS];
int rc;
long t;
initialize_memory();
pthread_mutex_init(&mutex, NULL);
for (t = 0; t < NUM_THREADS; t++) {
printf("Creating thread %ld\n", t);
rc = pthread_create(&threads[t], NULL, allocate_memory, (void *)&t);
if (rc) {
printf("ERROR: return code from pthread_create() is %d\n", rc);
exit(-1);
}
}
pthread_mutex_destroy(&mutex);
pthread_exit(NULL);
}
```

OUTPUT:




Develop a C program to do best fit, worst fit, first fit memory allocation of fixed partition with assumption of block sizes and processes memory request sizes are in process.txt file. So your program should read the data from the file and perform the memory allocations.

a) WORST CASE

CODE:

```
#include<stdio.h>
#define max 25
int main()
{
int frag[max],b[max],f[max],i,j,nb,nf,temp,highest=0;
static int bf[max],ff[max];
FILE *fp = fopen("a.txt","r");
fscanf(fp,"%d", &nb);
fscanf(fp,"%d",&nf);
for(i=1; i<=nb; i++){
fscanf(fp,"%d", &b[i]);
}
for(i=1;i<=nb;i++)
{
scanf("%d",&b[i]);
}
for(i=1;i<=nf;i++)
{
scanf("%d",&f[i]);
}
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(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:\tFragement");
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])
;
return 0;
}
```

b) BEST FIT

CODE:

```
#include<stdio.h>
#define max 25
int main()
{
int frag[max],b[max],f[max],i,j,nb,nf,temp,lowest=10000;
static int bf[max],ff[max];
FILE *fp = fopen("a.txt","r");
fscanf(fp,"%d", &nb);
fscanf(fp,"%d",&nf);
for(i=1; i<=nb; i++){
printf("Block %d:",i);
fscanf(fp,"%d", &b[i]);
}
printf("Enter the size of the files :-\n");
for(i=1; i<=nf; i++){
printf("FILE %d:",i);
fscanf(fp,"%d", &f[i]);
}
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>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]);
return 0;
}
```

c) FIRST FIT

CODE:

```
#include<stdio.h>
#define max 25
int main()
{
int frag[max],b[max],f[max],i,j,nb,nf,temp;
static int bf[max],ff[max];
FILE *fp = fopen("a.txt","r");
fscanf(fp,"%d", &nb);
fscanf(fp,"%d",&nf);
for(i=1; i<=nb; i++){
fscanf(fp,"%d", &b[i]);
}
for(i=1;i<=nf;i++)
{
scanf("%d",&f[i]);
}
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;
}
printf("\nFile_no:\tFile_size :\tBlock_no:\tBlock_size:\tFragement");
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]);
return 0;
}
```