NATIONAL INSTITUTE OF TECHNOLOGY KARNATAKA

OPERATING SYSTEM LAB

EXPERIMENT : 8

Name : Shivam Yadav Roll No : 211CS257

Section : S2 Code : CS257

Q1) Write a C program to simulate MVT and MFT memory management technique :

i)MVT:

```
#include<stdio.h>
int main()
{
    //totalmemory
        int tm;
        printf("\n Enter memory size : ");
        scanf("%d",&tm);

        //array for storing process size
        int a[100];

        //array for storing all process size
        int b[100];

        //start allocating
        int temp=tm;
        int i=0;
        int j=0;
        int total_allocated=0;
        while(temp > 0)
        {
                int ps;
                printf(" Enter Process %d size : ",j+1);
                scanf("%d",&ps);
                b[j++]=ps;
```

```
		if(ps<temp)
		{
			a[i++]=ps;
			temp=temp-ps;
		}
		else if(ps==temp)
		{
			a[i++]=ps;
			temp=0;
			break;
		}
		else
			break;
	}
	total_allocated=tm-temp;
	int p=tm;
	printf("\n Process No.\tRequired Space\tAvailable  \t  Status\n");
	for(int k=0;k<j;++k)
	{
		if(k<i)
		{
			printf("\n %d\t\t %d\t\t %d\t\t Allocated",k+1,b[k],p);
			p=p-b[k];
		}
		else
		{
			printf("\n %d\t\t %d\t\t %d\t\t Not Allocated",k+1,b[k],p);
		}
	}
```

```
        printf("\n");
        printf("\n total memory space : %d",tm);
        printf("\n total used memory : %d\n",total_allocated);
        printf(" External Fragmentation space : %d\n\n",temp);

}
```

output:

```
student@cclab-HP-EliteDesk-800-G1-TWR:~/Desktop$ gcc MVT.c
student@cclab-HP-EliteDesk-800-G1-TWR:~/Desktop$ ./a.out

 Enter memory size : 200
 Enter Process 1 size : 150
 Enter Process 2 size : 25
 Enter Process 3 size : 35

 Process No.    Required Space  Available         Status

 1               150             200             Allocated
 2               25              50              Allocated
 3               35              25              Not Allocated

 total memory space : 200
 total used memory : 175
 External Fragmentation space : 25
```

ii)MFT:

```
#include<stdio.h>
int main()
{
        //memory space
        int ms;
        printf("\n\nenter the memory space in bytes: ");
        scanf("%d",&ms);

        //size of each block
```

```
int bs;
printf("enter the size of each block in bytes: ");
scanf("%d",&bs);

//number of process
int n;
printf("enter the number of processes: ");
scanf("%d",&n);

//array to hold size of each process
int ps[n];
printf("Enter the memory space required by each process: ");
for(int i=0;i<n;i++)
        scanf("%d",&ps[i]);

//number of block
int nb;
nb=ms/bs;
printf("number of blocks: %d\n",nb);

//eval external fragmentation space
int ef;
ef=ms-(nb*bs);

//for internal fragmentation
int ifrag=0;
int i;
int j=0;
printf("\nProcess No \t Status \t\t Size of Internal Fragment\n");
for(i=0;i<n&&j<nb;i++)
{
```

```
                //printf("process %d: ",i+1);
                if(ps[i]<=bs)
                {
                        printf("%d\t\t Allocated \t\t %d\n",i+1,bs-ps[i]);
                        ifrag=ifrag+(bs-ps[i]);
                        j++;
                }
                else
                        printf("%d\t\t Not Allocated \n",i+1);
        }
        if(i<n-1)
                printf("remaining processes cannot be fit in memory.\n");
        printf("\nsize of internal fragment:%d\n",ifrag);
        printf("size of external fragment: %d\n\n",ef);

        return 0;
}
```

output:

```
student@cclab-HP-EliteDesk-800-G1-TWR:~/Desktop$ gcc MFT.c
student@cclab-HP-EliteDesk-800-G1-TWR:~/Desktop$ ./a.out


enter the memory space in bytes: 50
enter the size of each block in bytes: 15
enter the number of processes: 3
Enter the memory space required by each process: 12 13 15
number of blocks: 3

Process No      Status                  Size of Internal Fragment
1               Allocated               3
2               Allocated               2
3               Allocated               0

size of internal fragment:5
size of external fragment: 5
```

Q2) Write a C program to simulate First Fit , Best Fit and Worse Fit contiguous memory allocation techniques :

i) FIRST FIT :

```
#include<stdio.h>
int main()
{
        //number of memory block
        int nb;
        printf("\n enter no of memory block : ");
        scanf("%d",&nb);

        //number of process
        int np;
        printf("\n enter no of process : ");
        scanf("%d",&np);

        // block array
        int b[nb];
        printf("\n enter size for each memory block : ");
        for(int i=0;i<nb;++i)
                scanf("%d",&b[i]);

        //process array
        int p[np];
        printf("\n enter size of each process : ");
        for(int i=0;i<np;++i)
                scanf("%d",&p[i]);
        printf("\n Process No.\tProcess Size\tAllocated Block\n");
```

```
        for(int i=0;i<np;++i)
        {
                int k=-1;
                for(int j=0;j<nb;++j)
                {
                        if(p[i] <= b[j])
                        {
                                k=j;
                                b[j]=b[j]-p[i];
                                break;
                        }
                }
                if(k!=-1)
                        printf(" %d\t\t%d\t\t%d\n",i+1,p[i],k+1);
                else
                        printf(" %d\t\t%d\t\tNot Allocated\n",i+1,p[i]);
        }
        printf("\n");

}
```

output:

```
student@cclab-HP-EliteDesk-800-G1-TWR:~/Desktop$ gcc first_fit.c
student@cclab-HP-EliteDesk-800-G1-TWR:~/Desktop$ ./a.out

 enter no of memory block : 5

 enter no of process : 4

 enter size for each memory block : 200 600 300 400 700

 enter size of each process : 312 517 212 526

 Process No.    Process Size    Allocated Block
 1              312             2
 2              517             5
 3              212             2
 4              526             Not Allocated
```

ii) BEST FIT :

```
#include<stdio.h>
#define INT_MAX 2140000000
int main()
{
	//number of memory block
	int nb;
	printf("\n enter no of memory block : ");
	scanf("%d",&nb);
	//number of process
	int np;
	printf("\n enter no of process : ");
	scanf("%d",&np);

	// block array
	int b[nb];
	printf("\n enter size for each memory block : ");

	for(int i=0;i<nb;++i)
		scanf("%d",&b[i]);

	//process array
	int p[np];
	printf("\n enter size of each process : ");
	for(int i=0;i<np;++i)
		scanf("%d",&p[i]);

	printf("\n Process No.\tProcess Size\tAllocated Block\n");
	for(int i=0;i<np;++i)
	{
		int k=-1;
```

```
                int smallest=INT_MAX;
                for(int j=0;j<nb;++j)
                {
                        if(b[j]>=p[i] && smallest > b[j]-p[i])
                        {
                                smallest=b[j]-p[i];
                                k=j;
                        }
                }
                if(k!=-1)
                {
                        printf(" %d\t\t%d\t\t%d\n",i+1,p[i],k+1);
                        b[k]=b[k]-p[i];
                }
                else
                        printf(" %d\t\t%d\t\tNot Allocated\n",i+1,p[i]);
        }
        printf("\n");
}
```

Output:

```
student@cclab-HP-EliteDesk-800-G1-TWR:~/Desktop$ gcc best_fit.c
student@cclab-HP-EliteDesk-800-G1-TWR:~/Desktop$ ./a.out

 enter no of memory block : 5

 enter no of process : 4

 enter size for each memory block : 200 600 300 400 700

 enter size of each process : 312 517 212 526

 Process No.    Process Size    Allocated Block
 1              312             4
 2              517             2
 3              212             3
 4              526             5
```

iii) WORSE FIT:

```
#include<stdio.h>
#define INT_MIN -2140000000
int main()
{
        //number of memory block
        int nb;
        printf("\n enter no of memory block : ");
        scanf("%d",&nb);

        //number of process
        int np;
        printf("\n enter no of process : ");
        scanf("%d",&np);

        // block array
        int b[nb];
        printf("\n enter size for each memory block : ");

        for(int i=0;i<nb;++i)
                scanf("%d",&b[i]);

        //process array
        int p[np];
        printf("\n enter size of each process : ");
```

```
	for(int i=0;i<np;++i)
		scanf("%d",&p[i]);

	printf("\n Process No.\tProcess Size\tAllocated Block\n");

	for(int i=0;i<np;++i)
	{
		int k=-1;
		int bigger=INT_MIN;
		for(int j=0;j<nb;++j)
		{
			if(b[j]>=p[i] && bigger < b[j]-p[i])
			{
				bigger=b[j]-p[i];
				k=j;
			}
		}

		if(k!=-1)
		{
			printf(" %d\t\t%d\t\t%d\n",i+1,p[i],k+1);
			b[k]=b[k]-p[i];
		}
		else
			printf(" %d\t\t%d\t\tNot Allocated\n",i+1,p[i]);

	}
	printf("\n");
}
```

output:

```
student@cclab-HP-EliteDesk-800-G1-TWR:~/Desktop$ gcc worse_fit.c
student@cclab-HP-EliteDesk-800-G1-TWR:~/Desktop$ ./a.out

 enter no of memory block : 5

 enter no of process : 4

 enter size for each memory block : 200 600 300 400 700

 enter size of each process : 312 517 212 526

 Process No.    Process Size    Allocated Block
 1              312             5
 2              517             2
 3              212             4
 4              526             Not Allocated
```