21BAI1217
Mainak Chattopadhyay
OS LAB 10

1. Write a C program to implement memory allocation strategies.

CODE-

```
#include <stdio.h>

void implimentBestFit(int blockSize[], int blocks, int processSize[], int proccesses)
{
   // This will store the block id of the allocated block to a process
   int allocation[proccesses];
   int occupied[blocks];

   // initially assigning -1 to all allocation indexes
   // means nothing is allocated currently
   for(int i = 0; i < proccesses; 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 < proccesses; i++)
   {

      int indexPlaced = -1;
      for (int j = 0; j < blocks; j++) {
         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 smalller than the current block where
            // process is placed, change the block and thus indexPlaced
                  // this reduces the wastage thus best fit
            else if (blockSize[j] < blockSize[indexPlaced])
               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;
        }
    }
    printf("\n Best fit allocation\n");
    printf("\nProcess No.\tProcess Size\tBlock no.\n");
    for (int i = 0; i < proccesses; 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");
    }
}
void implementFirstFit(int bsize[], int bno, int psize[], int pno,int allocation[],int flags[]){
for(int i = 0; i < pno; i++) { //allocation as per first fit
for(int j = 0; j < bno; j++)
if(flags[j] == 0 && bsize[j] >= psize[i])
{
allocation[j] = i;
flags[j] = 1;
break;
}
}
        //display allocation details
        printf("\n First fit allocation \n");
        printf("\nBlock no.\tsize\t\tprocess no.\t\tsize");
        for(int 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\n",allocation[i]+1,psize[allocation[i]]);
                else
                        printf("Not allocated \n");
        }
}

int worstfit(int b[], int nb, int f[], int nf){
```

```
	printf("\n\n\t\t\tMemory Management Scheme - Worst Fit");
	int frag[25],i,j,temp,highest=0;
	static int bf[25],ff[25];
	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("\nProcess_no \tProcess_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]);
	printf("\n");
}

// Driver code
int 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: \n");
	scanf("%d", &bno);
	printf("\nEnter size of each block: \n");
	for(i = 0; i < bno; i++)
		scanf("%d", &bsize[i]);
	printf("\nEnter no. of processes: \n");
	scanf("%d", &pno);
	printf("\nEnter size of each process: \n");
	for(i = 0; i < pno; i++)
```

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

    implimentBestFit(bsize, bno, psize, pno);
    implementFirstFit(bsize, bno, psize, pno,allocation,flags);
    worstfit(bsize, bno, psize, pno);

    return 0 ;
}
```

OUTPUT

```
ex2@AB1205BSCS08:~$ gedit lab10.c
ex2@AB1205BSCS08:~$ gcc lab10.c
ex2@AB1205BSCS08:~$ ./a.out
Enter no. of blocks:
5

Enter size of each block:
100
250
200
350
500

Enter no. of processes:
5

Enter size of each process:
150
90
250
200
190
```

```
 Best fit allocation

Process No.     Process Size    Block no.
1                       150                     3
2                       90                      1
3                       250                     2
4                       200                     4
5                       190                     5
```

First fit allocation

Block no.	size	process no.	size
1	100	2	90
2	250	1	150
3	200	4	200
4	350	3	250
5	500	5	190

Memory Management Scheme - Worst Fit

Process_no	Process_size	Block_no	Block_size	Fragment
1	90	4	500	410
2	250	3	350	100
3	200	1	250	50
4	190	2	200	10
5	0	0	100	0