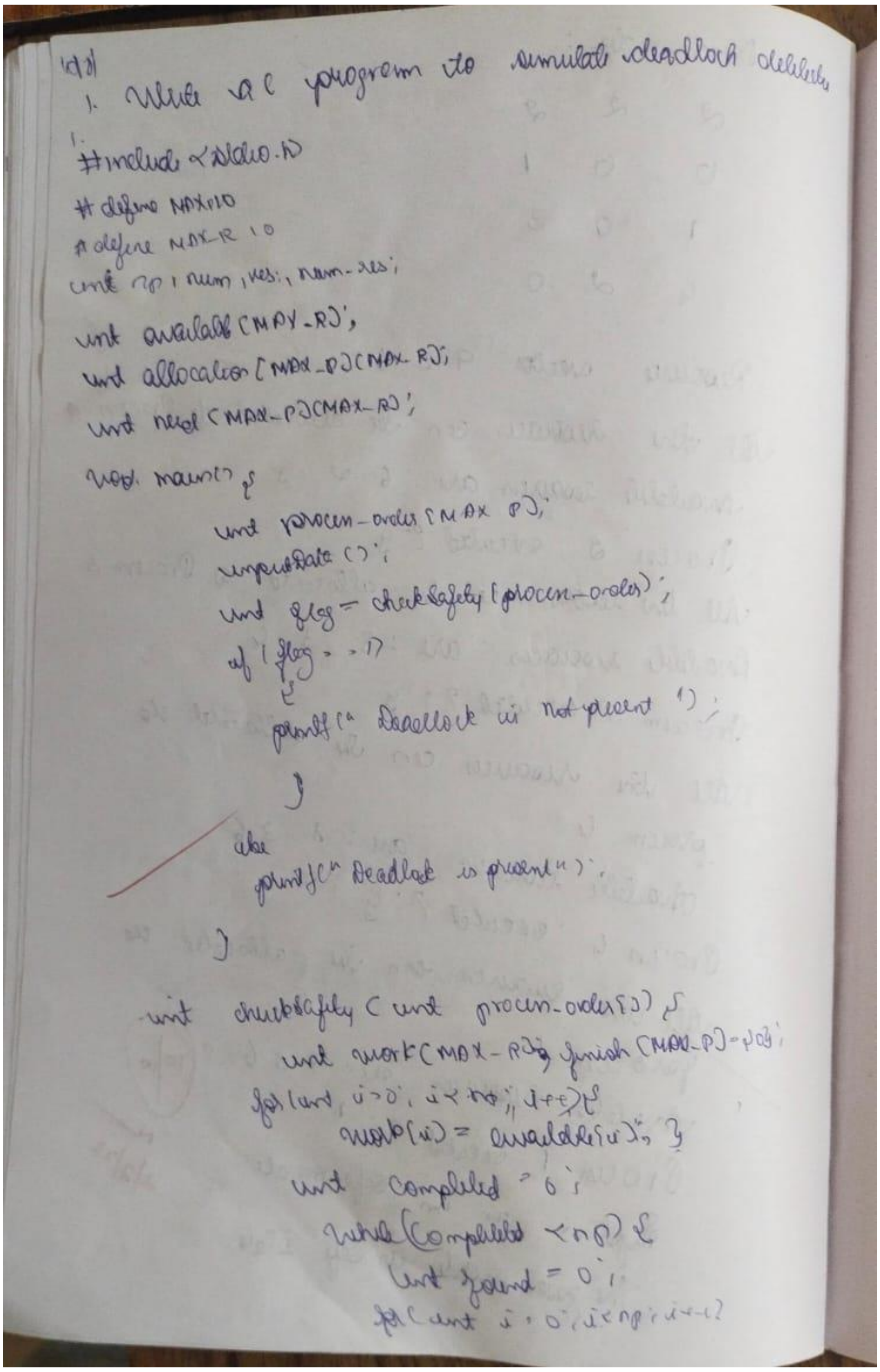
OS LAB-7

Q) **a) Write a C program to simulate deadlock detection**

b) Write a C program to simulate the following contiguous memory allocation techniques a) Worst-fit b) Best-fit c) First-fit

CODE-



```
1. Write a C program to simulate deadlock detection

#include <stdio.h>
# define MAX_P 10
# define MAX_R 10
int np, num_res, num_res;
int available[MAX_R];
int allocation[MAX_P][MAX_R];
int need[MAX_P][MAX_R];

void main() {
        int process_order[MAX_P];
        inputData();
        int flag = checkSafety(process_order);
        if (flag == 1)
        {
          printf(" Deadlock is not present ");
          }
        else
        printf(" Deadlock is present ");
    }

int checkSafety(int process_order[]) {
      int work[MAX_R], finish[MAX_P] = {0};
    for(int i=0; i<np; i++){
          work[i] = available[i]; }
       int completed = 0;
        while(completed < np) {
          int found = 0;
          for(int i = 0; i<np; i++)
    {.   if (!finish[i]){
              int j;
           for (j = 0; j < num_res ; j++){
                if (need[i][j] > work [j]){
                          break; }}

        if (j == num_res){
                 for( int k = 0; k < num_res ; k++)
         {
             work [k]+ = allocation [i][k]; }

         finish[i] = 1;
         process_order [completed] = i ;
         completed ++; found = 1; } } }
         if (! found ) { return 0; } } }
         return 1;

      }
void inputData () {
      printf("Enter the number of process, resources, available
          resources and request matrix");
      scanf("%d", &np, & num_res);
         for (int i = 0; i < num_res; i++){
                scanf("%d", &available [i]); }
         for (int i = 0; ~~[illegible]~~ i < np; i++){
            for (int j = 0; j < num_res; j++){
                   scanf ("%d", &max[i][j]);

                  }
  printf("enter the allocation matrix:\n");
         for (int i = 0; i < np; i++){
              for (int j = 0; j < nr; j++)
           { scanf("%d", & allocation[i][j]);
                 need[i][j] = max[i][j] - allocation[i][j];
                } } }
```

Write a C Program to simulate the following contiguous memory allocation using
a) Worst-fit b) Best-fit c) First fit.

a) Worst fit

```
#include<stdio.h>
void worstfit (int blockSize[], int blocks, int processSize[],
               int processes)
{
    int allocation[processes], occupied[blocks];
    for(int i=0; i<processes; i++){
            allocation[i]=-1; }
  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[indexPlaced] < blockSize[indexPlaced])
             indexPlaced = j;
   }
   }
    if (indexPlaced != -1){
         allocation[i] = indexPlaced;
         occupied[indexPlaced] = 1;
         blockSize[indexPlaced] -= processSize[i];
         }}
 printf("\nProcess No.\tProcess Size\tBlock no\n");
for (int i = 0; i < processes; i++)
{
 printf("%d \t %d \t %d \t\t", i+1, processSize[i]);
 if (allocation[i] != -1)
    printf("%d \n", allocation[i]+1);
   else
    printf("Not allocated\n");
}}

int main()
 {
   int i, blocks, processes;
   printf("Enter the no of blocks :");
   scanf("%d", &blocks);
   int blocksize[blocks];
   printf("Enter the size of each block:");
   for(int i=0; i<blocks; i++)
   scanf("%d", &processes);
    int processSize[processes];
   printf("Enter size of each process:");
   for (i=0; i<processes; i++)
      scanf("%d", &processSize[i]);
   worstfit (blocksize, blocks, processSize, processes);
   return 0;

 }
```

Output

```
Enter no of blocks: 3
Enter size of each block: 5 2 7
 Enter no of process: 2
Enter size of each process : 1 4
```

Process NO	Process size	Block no
1	1	3
2	4	1

```
2) #include <stdio.h>
#define MAX 10
void BestFit (int blocksize[] , int blocks, int processsize[],
        int processes, int m) {
    int allocation [processes], occupied [blocks];
    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 (indexPlaced != -1)
    {
      allocation [i] = indexPlaced ;
      occupied [indexPlaced] = 1;
    }
  }
printf("\nProcess No\t Process Size \t Block no\n");
for (int i=0; i< processes; 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");

    }
}

int main(){
    int p,m,j;
    printf("Enter the number of process & blocks\n");
    scanf("%d%d", &p, &m);
    int processSize[p], blockSize[m];
    printf("Enter the process sizes: \n");
    for(j=0;j<p;j++)
        scanf("%d", &processSize[j]);
    printf("Enter the Block sizes:\n");
      for(j=0;j<m;j++) {
        scanf("%d", & blockSize[j]); } .
    int blocks = sizeof(BlockSize)/sizeof(blockSize[0]);
    int process = sizeof(processSize)/sizeof(processSize[0]);
    Bestfit(blockSize, blocks, processSize, process, m);
    return 0;
    }
```

Output

```
Enter the number of process and blocks: 2 3
Enter the process size: 1 4
Enter the block size: 5 2 7
Process No    Process Size    Block no
   1              1              2
   2              4              1
```

c) First fit

```
# include < stdio.h>
#include <conio.h>
# define max 25
void main() {
    int frg[max], b[max], f[max], i, j, nb, nf, temp;
    static int bf[max], ff[max];
    printf("Memory management scheme - First Fit");
    printf("Enter the number of blocks :");
    scanf("%d", &nb);
    printf("Enter the number of files :");
    scanf("%d", &nf);
    printf("Enter the size of block -\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]); }
    printf("\n File no: \t File size : \t Block no. \t Block-size\t");
      for (i=1; i<=nf; i++) {
                int allocated = 0;
            for (j=1; j<=nb; j++) {
               if (bf[j] != 1) {
                   temp = b[j]-f[i];
               if (temp >= 0) {
                   ff[i] = j;
                     bf[j]=1;
                 frag[i] = b[j]-f[i];
                   allocated = 1;
printf("\n%d\t\t%d\t\t%d\t\t%d", i+1, f[i], ff[i],
        b[ff[i]]);
        break;
    }
}
if (!allocated){
    printf("\n%d\t\t%d\t\t Not allocated\t\t-", i+1, f[i]);
    }
  }
  getch();
}.
```

Output

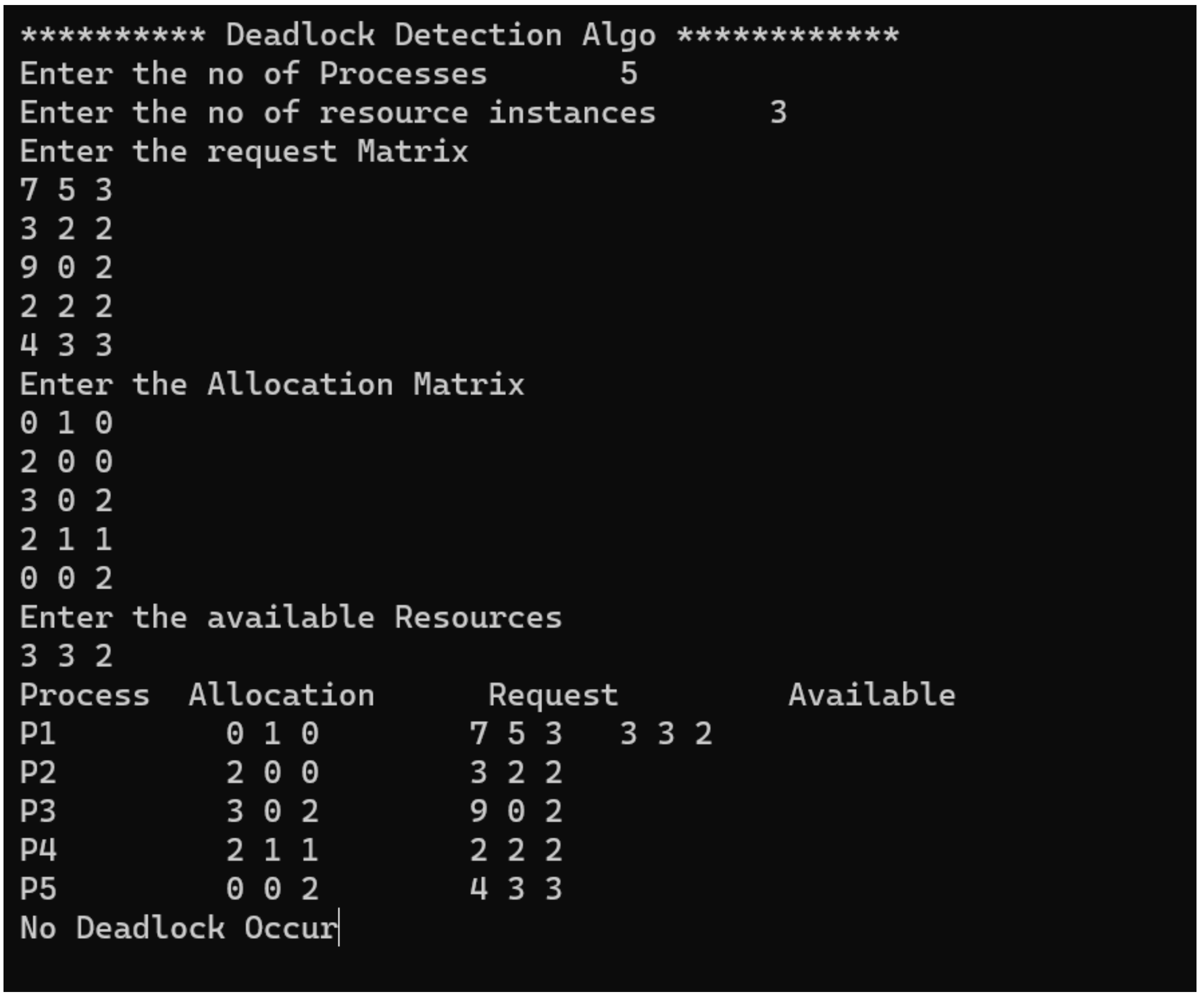
```
Memory Management scheme - First Fit

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
  1          1           1          5
  2          4           3          7.
```

10/10

23/8/23

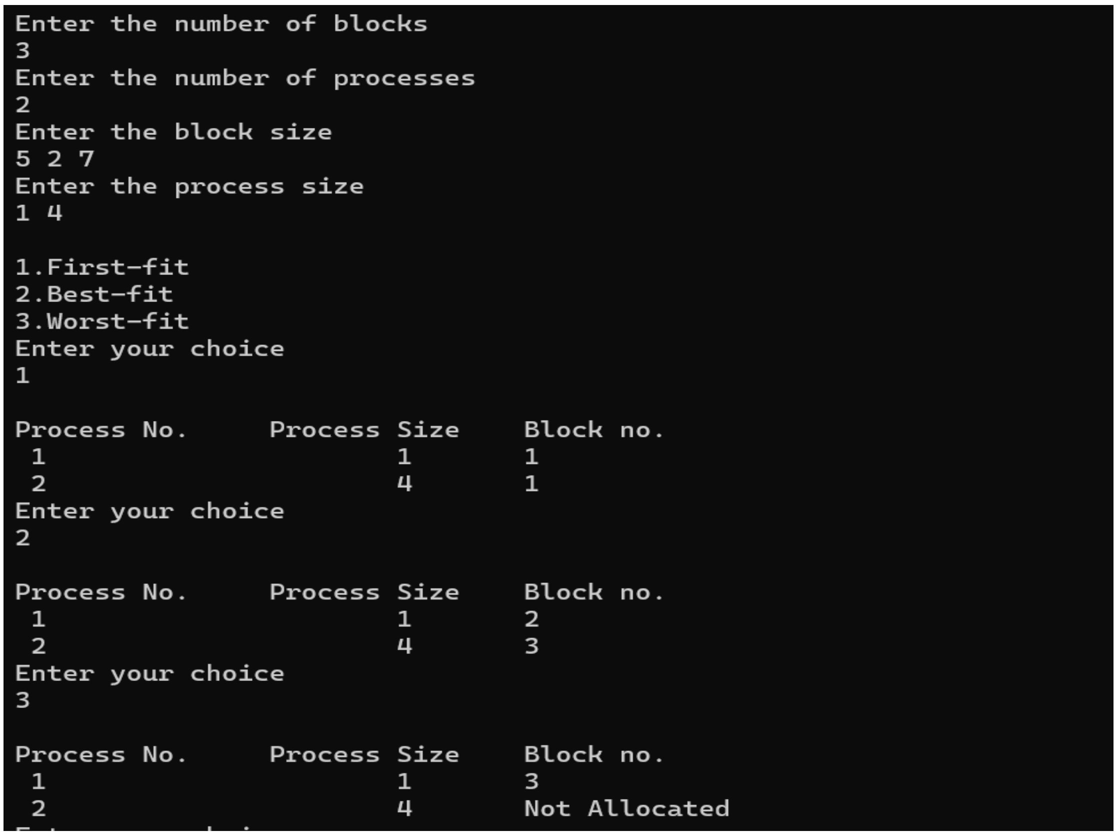
OUTPUT-

Deadlock Detection



```
********** Deadlock Detection Algo ************
Enter the no of Processes       5
Enter the no of resource instances      3
Enter the request Matrix
7 5 3
3 2 2
9 0 2
2 2 2
4 3 3
Enter the Allocation Matrix
0 1 0
2 0 0
3 0 2
2 1 1
0 0 2
Enter the available Resources
3 3 2
Process  Allocation      Request         Available
P1          0 1 0        7 5 3   3 3 2
P2          2 0 0        3 2 2
P3          3 0 2        9 0 2
P4          2 1 1        2 2 2
P5          0 0 2        4 3 3
No Deadlock Occur
```

Contiguous Allocation



```
Enter the number of blocks
3
Enter the number of processes
2
Enter the block size
5 2 7
Enter the process size
1 4

1.First-fit
2.Best-fit
3.Worst-fit
Enter your choice
1

Process No.     Process Size    Block no.
 1                      1       1
 2                      4       1
Enter your choice
2

Process No.     Process Size    Block no.
 1                      1       2
 2                      4       3
Enter your choice
3

Process No.     Process Size    Block no.
 1                      1       3
 2                      4       Not Allocated
```