KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY (KIIT)

Deemed to be University U/S 3 of UGC Act, 1956

OS LAB ASSIG 10

- Name :HITU RAJ
- Roll no. :2005025
- Branch :CSE

2005025_Hitu raj

1.BANKERS ALGORITHM

```
1.

// BANKERS ALGORITHM

#include <stdio.h>
#include <conio.h>

int main()
{
    int Max[10][10], need[10][10], alloc[10][10], avail[10], completed[10],
safeSequence[10];
    int p, r, i, j, process, count;
    count = 0;

    printf("Enter the no of processes : ");
    scanf("%d", &p);

    for (i = 0; i < p; i++)
```

```
        completed[i] = 0;

    printf("\n\nEnter the no of resources : ");
    scanf("%d", &r);

    printf("\n\nEnter the Max Matrix for each process : ");
    for (i = 0; i < p; i++)
    {
        printf("\nFor process %d : ", i + 1);
        for (j = 0; j < r; j++)
            scanf("%d", &Max[i][j]);
    }

    printf("\n\nEnter the allocation for each process : ");
    for (i = 0; i < p; i++)
    {
        printf("\nFor process %d : ", i + 1);
        for (j = 0; j < r; j++)
            scanf("%d", &alloc[i][j]);
    }

    printf("\n\nEnter the Available Resources : ");
    for (i = 0; i < r; i++)
        scanf("%d", &avail[i]);

    for (i = 0; i < p; i++)
        for (j = 0; j < r; j++)
            need[i][j] = Max[i][j] - alloc[i][j];

    do
    {
        printf("\n Max matrix:\tAllocation matrix:\n");
        for (i = 0; i < p; i++)
        {
            for (j = 0; j < r; j++)
                printf("%d  ", Max[i][j]);
            printf("\t\t");
            for (j = 0; j < r; j++)
                printf("%d  ", alloc[i][j]);
            printf("\n");
        }

        process = -1;

        for (i = 0; i < p; i++)
        {
            if (completed[i] == 0) // if not completed
            {
                process = i;
                for (j = 0; j < r; j++)
                {
                    if (avail[j] < need[i][j])
                    {
                        process = -1;
```

```
                        break;
                    }
                }
            }
            if (process != -1)
                break;
        }

        if (process != -1)
        {
            printf("\nProcess %d runs to completion!", process + 1);
            safeSequence[count] = process + 1;
            count++;
            for (j = 0; j < r; j++)
            {
                avail[j] += alloc[process][j];
                alloc[process][j] = 0;
                Max[process][j] = 0;
                completed[process] = 1;
            }
        }
    } while (count != p && process != -1);

    if (count == p)
    {
        printf("\nThe system is in a safe state!!\n");
        printf("Safe Sequence : < ");
        for (i = 0; i < p; i++)
            printf("%d  ", safeSequence[i]);
        printf(">\n");
    }
    else
        printf("\nThe system is in an unsafe state!!");
    getch();
}
```

OUTPUT

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS D:\my codes\hackerrank> cd "d:\my codes\hackerrank\" ; if ($?) { g++ BANKER.C -o BANKER } ; if ($?) { .\BANKER }
Enter the no of processes : 5


Enter the no of resources : 3


Enter the Max Matrix for each process :
For process 1 : 4
3
3

For process 2 : 3
2
2

For process 3 : 9
0
2

For process 4 : 7
5
1

For process 5 : 1
1
2


Enter the allocation for each process :
For process 1 : 2
For process 1 : 2
1
2

For process 2 : 4
0
1

For process 3 : 0
20
0

For process 4 : 2
1
3

For process 5 : 12
3
2


Enter the Available Resources : 2
1
0

 Max matrix:    Allocation matrix:
4  3  3                2  1  2
3  2  2                4  0  1
9  0  2                0  20  0
7  5  1                2  1  3
1  1  2                12  3  2

9  0  2                0  20  0
7  5  1                2  1  3
0  0  0                0  0  0
```

```
9  0  2               0  20  0
7  5  1               2  1  3
0  0  0               0  0  0

Process 2 runs to completion!
 Max matrix:    Allocation matrix:
0  0  0               0  0  0
0  0  0               0  0  0
9  0  2               0  20  0
7  5  1               2  1  3
0  0  0               0  0  0

Process 3 runs to completion!
 Max matrix:    Allocation matrix:
0  0  0               0  0  0
0  0  0               0  0  0
0  0  0               0  0  0
7  5  1               2  1  3
0  0  0               0  0  0

Process 4 runs to completion!
The system is in a safe state!!
Safe Sequence : < 5  1  2  3  4  >
```

2.FIRST FIT

```c
// FIRST FIT:

#include <stdio.h>
void 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: ");
    scanf("%d", &bno);
    printf("\nEnter size of each block: ");
    for (i = 0; i < bno; i++)
        scanf("%d", &bsize[i]);

    printf("\nEnter no. of processes: ");
    scanf("%d", &pno);
    printf("\nEnter size of each process: ");
    for (i = 0; i < pno; i++)
        scanf("%d", &psize[i]);
    for (i = 0; i < pno; i++) // allocation as per first fit
        for (j = 0; j < bno; j++)
            if (flags[j] == 0 && bsize[j] >= psize[i])
            {
                allocation[j] = i;
```

```
                flags[j] = 1;
                break;
            }
    // display allocation details
    printf("\nBlock no.\tsize\t\tprocess no.\t\tsize");
    for (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", allocation[i] + 1, psize[allocation[i]]);
        else
            printf("Not allocated");
    }
}
```

OUTPUT

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL                                   Code

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS D:\my codes\hackerrank> cd "d:\my codes\hackerrank\" ; if ($?) { g++ FIRST.C -o FIRST } ; if ($?) { .\FIRST }
Enter no. of blocks: 4

Enter size of each block: 400
600
1000
800

Enter no. of processes: 4

Enter size of each process: 500
450
712
385

Block no.       size            process no.     size
1               400             4               385
2               600             1               500
3               1000            2               450
4               800             3               712
PS D:\my codes\hackerrank>
```

3.BEST FIT

```
// BEST FIT:

#include <stdio.h>

int main()
{
    int fragment[20], b[20], p[20], i, j, nb, np, temp, lowest = 9999;
    static int barray[20], parray[20];
```

```
    printf("\n\t\t\tMemory Management Scheme - Best Fit");
    printf("\nEnter the number of blocks:");
    scanf("%d", &nb);
    printf("Enter the number of processes:");
    scanf("%d", &np);
    printf("\nEnter the size of the blocks:-\n");
    for (i = 1; i <= nb; i++)
    {
        printf("Block no.%d:", i);
        scanf("%d", &b[i]);
    }
    printf("\nEnter the size of the processes :-\n");
    for (i = 1; i <= np; i++)
    {
        printf("Process no.%d:", i);
        scanf("%d", &p[i]);
    }
    for (i = 1; i <= np; i++)
    {
        for (j = 1; j <= nb; j++)
        {
            if (barray[j] != 1)
            {
                temp = b[j] - p[i];
                if (temp >= 0)
                    if (lowest > temp)
                    {
                        parray[i] = j;
                        lowest = temp;
                    }
            }
        }
        fragment[i] = lowest;
        barray[parray[i]] = 1;
        lowest = 10000;
    }
    printf("\nProcess_no\tProcess_size\tBlock_no\tBlock_size\tFragment");
    for (i = 1; i <= np && parray[i] != 0; i++)
        printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d", i, p[i], parray[i], b[parray[i]],
fragment[i]);
        return 0;
}
```

OUTPUT

```
Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS D:\my codes\hackerrank> cd "d:\my codes\hackerrank\" ; if ($?) { g++ BEST.C -o BEST } ; if ($?) { .\BEST }

                        Memory Management Scheme - Best Fit
Enter the number of blocks:4
Enter the number of processes:4

Enter the size of the blocks:-
Block no.1:400
Block no.2:600
Block no.3:1000
Block no.4:800

Enter the size of the processes :-
Process no.1:500
Process no.2:450
Process no.3:750
Process no.4:350

Process_no      Process_size    Block_no        Block_size      Fragment
1               500             2               600             100
2               450             4               800             350
3               750             3               1000            250
4               350             1               400             50
PS D:\my codes\hackerrank>
```

4.WORST FIT

```c
// WORST FIT:

#include <stdio.h>

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

    // initially assigning -1 to all allocation indexes
    // means nothing is allocated currently
    for (int i = 0; i < processes; 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 < processes; i++)
    {
        int indexPlaced = -1;
        for (int j = 0; j < blocks; j++)
        {
            // if not occupied and block size is large enough
            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 larger than the current block where
                // process is placed, change the block and thus indexPlaced
                else if (blockSize[indexPlaced] < blockSize[j])
                    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;

            // Reduce available memory for the block
            blockSize[indexPlaced] -= processSize[i];
        }
    }

    printf("\nProcess No.\tProcess Size\tBlock 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");
    }
}

// Driver code
int main()
{
    int n;
    printf("Enter no of blocks:");
    scanf("%d", &n);
    int blockSize[n];
    int m;
    printf("Enter no of process:");
    scanf("%d", &m);
```

```
    int processSize[m];
    for (int i = 0; i < n; i++)
    {
        printf("Enter size for block %d ", i + 1);
        scanf("%d", &blockSize[i]);
    }
    for (int i = 0; i < n; i++)
    {
        printf("Enter size for process no %d ", i + 1);
        scanf("%d", &processSize[i]);
    }

    int blocks = sizeof(blockSize) / sizeof(blockSize[0]);
    int processes = sizeof(processSize) / sizeof(processSize[0]);

    implimentWorstFit(blockSize, blocks, processSize, processes);

    return 0;
}
```

OUTPUT