OPERATING SYSTEMS

Lab 11: First Fit, Best Fit, Worst Fit - Variable Partitioning, First Fit - Fixed Partitioning

Name – Vinay Santosh Menon
Registration Number - 20BAI1103

TASK:

To write a C code for First Fit, Best Fit, Worst Fit - Variable Partitioning, First Fit - Fixed Partitioning

FIRST FIT (Fixed Partitioning):

Code:

```
#include <stdio.h>

void implimentFirstFit(int blockSize[], int blocks, int processSize[], int
processes)
{
    int allocate[processes];
    int occupied[blocks];

    for(int i = 0; i < processes; i++)
    {
        allocate[i] = -1;
    }

    for(int i = 0; i < blocks; i++){
        occupied[i] = 0;
    }

    for (int i = 0; i < processes; i++)
    {
        for (int j = 0; j < blocks; j++)
        {
        if (!occupied[j] && blockSize[j] >= processSize[i])
            {
                allocate[i] = j;
                occupied[j] = 1;
                break;
            }
        }
    }

    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 (allocate[i] != -1)
            printf("%d\n",allocate[i] + 1);
        else
            printf("Not Allocated\n");
    }
}
```

```
void main()
{
    int blockSize[] = {30, 5, 10};
    int processSize[] = {10, 6, 9};
    int m = sizeof(blockSize)/sizeof(blockSize[0]);
    int n = sizeof(processSize)/sizeof(processSize[0]);

    implimentFirstFit(blockSize, m, processSize, n);
}
```

Output:

```
Process No.     Process Size    Block no.
1               10              1
2               6               3
3               9               Not Allocated
```

FIRST FIT (VARIABLE PARTITIONING):

Code:

```
#include <stdio.h>

void implimentFirstFit(int blockSize[], int blocks, int processSize[], int
processes)
{
    int allocate[processes];

    for(int i = 0; i < processes; i++)
    {
        allocate[i] = -1;
    }

    for (int i = 0; i < processes; i++)
    {
        for (int j = 0; j < blocks; j++) {
            if (blockSize[j] >= processSize[i])
            {

                allocate[i] = j;

                blockSize[j] -= processSize[i];

                break;
            }
        }
```

```
    }

    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 (allocate[i] != -1)
            printf("%d\n",allocate[i] + 1);
        else
            printf("Not Allocated\n");
    }
}
void main()
{
    int blockSize[] = {15, 12, 10};
    int processSize[] = {10, 12,5};
    int m = sizeof(blockSize)/sizeof(blockSize[0]);
    int n = sizeof(processSize)/sizeof(processSize[0]);

    implimentFirstFit(blockSize, m, processSize, n);
}
```

Output:

```
Process No.     Process Size    Block no.
1                       10                      1
2                       12                      2
3                       5                       1
```

BEST FIT (FIXED PARTITIONING):

Code:

```
#include <stdio.h>

void BestFit(int blockSize[], int blocks, int processSize[], int proccesses)
{

    int allocation[proccesses];
    int occupied[blocks];

    for(int i = 0; i < proccesses; i++){
        allocation[i] = -1;
```

```
    }

    for(int i = 0; i < blocks; i++){
        occupied[i] = 0;
    }

    for (int i = 0; i < proccesses; 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.\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");
    }
}

int main()
{
    int blockSize[] = {80, 50, 30, 120, 35};
    int processSize[] = {110, 10, 30, 60};
    int blocks = sizeof(blockSize)/sizeof(blockSize[0]);
    int proccesses = sizeof(processSize)/sizeof(processSize[0]);

    BestFit(blockSize, blocks, processSize, proccesses);
```

```
    return 0 ;
}
```

Output:

```
Process No.     Process Size    Block no.
1                       110                     4
2                       10                      3
3                       30                      5
4                       60                      1
```

BEST FIT (VARIABLE PARTITIONING):

Code:

```
#include <stdio.h>

void BestFit(int blockSize[], int blocks, int processSize[], int processes)
{
    int allocation[processes];

    for(int i = 0; i < processes; i++){
        allocation[i] = -1;
    }

    for (int i=0; i < processes; i++)
    {

        int indexPlaced = -1;
        for (int j=0; j < blocks; j++)
        {
            if (blockSize[j] >= processSize[i])
            {
                if (indexPlaced == -1)
                    indexPlaced = j;
                else if (blockSize[j] < blockSize[indexPlaced])
                    indexPlaced = j;
            }
        }

        if (indexPlaced != -1)
        {
            allocation[i] = indexPlaced;
            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");
    }
}

int main()
{
    int blockSize[] = {80, 50, 30, 120, 35};
    int processSize[] = {110, 10, 30, 60};
    int blocks = sizeof(blockSize)/sizeof(blockSize[0]);
    int processes = sizeof(processSize)/sizeof(processSize[0]);

    BestFit(blockSize, blocks, processSize, processes);

    return 0 ;
}
```

Output:

```
Process No.     Process Size    Block no.
1                       110                     4
2                       10                      4
3                       30                      3
4                       60                      1
```

WORST FIT (FIXED PARTITIONING)

Code:

```
#include <stdio.h>

void implimentWorstFit(int blockSize[], int blocks, int processSize[], int
processes)
{
    int allocation[processes];
    int 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[j])
                    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\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 blockSize[] = {80, 50, 130, 20, 35};
```

```
    int processSize[] = {30, 10, 50, 60};
    int blocks = sizeof(blockSize)/sizeof(blockSize[0]);
    int processes = sizeof(processSize)/sizeof(processSize[0]);

    implimentWorstFit(blockSize, blocks, processSize, processes);

    return 0;
}
```

Output:

```
Process No.     Process Size    Block no.
1               30              3
2               10              1
3               50              2
4               60              Not Allocated
```

WORST FIT (Variable Partitioning):

Code:

```
#include <stdio.h>

void implimentWorstFit(int blockSize[], int blocks, int processSize[], int
processes)
{
    int allocation[processes];

    for(int i = 0; i < processes; i++){
        allocation[i] = -1;
    }

    for (int i=0; i<processes; i++)
    {
        int indexPlaced = -1;
        for (int j=0; j<blocks; j++)
        {
            if (blockSize[j] >= processSize[i])
            {
                if (indexPlaced == -1)
                    indexPlaced = j;
                else if (blockSize[indexPlaced] < blockSize[j])
                    indexPlaced = j;
            }
        }
        if (indexPlaced != -1)
```

```
        {
            allocation[i] = indexPlaced;
            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");
    }
}

int main()
{
    int blockSize[] = {80, 50, 130, 20, 35};
    int processSize[] = {30, 10, 50, 60};
    int blocks = sizeof(blockSize)/sizeof(blockSize[0]);
    int processes = sizeof(processSize)/sizeof(processSize[0]);

    implimentWorstFit(blockSize, blocks, processSize, processes);

    return 0 ;
}
```

Output:

```
Process No.     Process Size    Block no.
1                       30                      3
2                       10                      3
3                       50                      3
4                       60                      1
```