MEMORY_MANAGEMENT

BCSE303P Operating Systems

Name: Chillara V L N S Pavana Vamsi
Register Number: 21BCE5095

1. Develop a C program to do best fit, worst fit, first fit memory allocation of fixed partition.

Code:

```c
#include <stdio.h>
void firstFit(int process_size[], int m, int space[], int n)
{
    printf("\n\n\t\tFirst fit");
    int allocation[m];
    for (int i = 0; i < m; i++)
        allocation[i] = -1;
    printf("\nProcess id\tProcess Size\tBlock No.\n");
    for (int i = 0; i < m; i++)
    {
        for (int j = 0; j < n; j++)
        {
            if (space[j] >= process_size[i])
            {
                space[j] -= process_size[i];
                allocation[i] = j;
                break;
            }
        }
        printf("%d\t\t%d\t\t", i + 1, process_size[i]);
        if (allocation[i] == -1)
            printf("Not allocated\n");
        else
            printf("%i\n", allocation[i] + 1);
    }
}
void bestFit(int process_size[], int m, int space[], int n)
{
    printf("\n\n\t\tBest fit");
    int allocation[m];
    for (int i = 0; i < m; i++)
        allocation[i] = -1;
    printf("\nProcess id\tProcess Size\tBlock No.\n");
    for (int i = 0; i < m; i++)
    {
        int x = -1;
        for (int j = 0; j < n; j++)
        {

            if (space[j] >= process_size[i])
            {
                if (x == -1)
                    x = j;
                else if (space[x] > space[j])
                    x = j;
            }
        }
        if (x != -1)
        {
```

```
            allocation[i] = x;
            space[x] -= process_size[i];
        }
        printf("%d\t\t%d\t\t", i + 1, process_size[i]);
        if (allocation[i] == -1)
            printf("Not allocated\n");
        else
            printf("%i\n", allocation[i] + 1);
    }
}
void worstFit(int process_size[], int m, int space[], int n)
{
    printf("\n\n\t\tWorst fit");
    int allocation[m];
    for (int i = 0; i < m; i++)
        allocation[i] = -1;
    printf("\nProcess id\tProcess Size\tBlock No.\n");
    for (int i = 0; i < m; i++)
    {
        int x = -1;
        for (int j = 0; j < n; j++)
        {

            if (space[j] >= process_size[i])
            {
                if (x == -1)
                    x = j;
                else if (space[x] < space[j])
                    x = j;
            }
        }
        if (x != -1)
        {
            allocation[i] = x;
            space[x] -= process_size[i];
        }
        printf("%d\t\t%d\t\t", i + 1, process_size[i]);
        if (allocation[i] == -1)
            printf("Not allocated\n");
        else
            printf("%i\n", allocation[i] + 1);
    }
}
int main()
{
    int n;
    printf("Enter the number of sections memory of 10GB divided:");
    scanf("%d", &n);
    int space1[n];
    int space2[n];
    int space3[n];
    for (int i = 0; i < n; i++)
    {
        printf("\nEnter the divison number %d (in GB):", i + 1);
```

```
        scanf("%d", &space1[i]);
        space2[i] = space1[i];
        space3[i] = space1[i];
    }

    int m;
    printf("Enter the number process:");
    scanf("%d", &m);
    int process_size[m];

    for (int i = 0; i < m; i++)
    {
        printf("\nEnter the allocation required for process number %d (in GB):", i + 1);
        scanf("%d", &process_size[i]);
    }

    firstFit(process_size, m, space1, n);
    bestFit(process_size, m, space2, n);
    worstFit(process_size, m, space3, n);
    return 0;
}
```

Output:

```
Enter the number of sections memory of 10GB divided:5

Enter the divison number 1 (in GB):100

Enter the divison number 2 (in GB):500

Enter the divison number 3 (in GB):200

Enter the divison number 4 (in GB):300

Enter the divison number 5 (in GB):600
Enter the number process:4

Enter the allocation required for process number 1 (in GB):212

Enter the allocation required for process number 2 (in GB):417

Enter the allocation required for process number 3 (in GB):112

Enter the allocation required for process number 4 (in GB):426
```

```
                First fit
Process id      Process Size    Block No.
1               212             2
2               417             5
3               112             2
4               426             Not allocated


                Best fit
Process id      Process Size    Block No.
1               212             4
2               417             2
3               112             3
4               426             5


                Worst fit
Process id      Process Size    Block No.
1               212             5
2               417             2
3               112             5
4               426             Not allocated
PS D:\VIT\Sem 2-2\OPERATING SYSTEMS\Assignments\lab 12>
```

2. Develop a C program to do best fit, worst fit, first fit memory allocation of fixed partition using 3 threads and and threads may return the name of the processes which are unallocated if any.

Code:

```c
#include <stdio.h>
#include <pthread.h>
#define n 5
#define m 4
struct processData
{
    int space[n];
    int process_size[m];
    int allocation[m];
    int unallocated[m];
};
void *firstFit(void *p)
{
    struct processData *pd = (struct processData *)p;
    printf("\n\n\t\tFirst fit");

    for (int i = 0; i < m; i++)
        pd->allocation[i] = -1;
    printf("\nProcess id\tProcess Size\tBlock No.\n");
    for (int i = 0; i < m; i++)
    {
        for (int j = 0; j < n; j++)
        {
            if (pd->space[j] >= pd->process_size[i])
            {
                pd->space[j] -= pd->process_size[i];
```

```
                pd->allocation[i] = j;
                break;
            }
        }
        printf("%d\t\t%d\t\t", i + 1, pd->process_size[i]);
        if (pd->allocation[i] == -1)
        {
            printf("Not allocated\n");
        }

        else
            printf("%i\n", pd->allocation[i] + 1);
    }
}
void *bestFit(void *p)
{
    struct processData *pd = (struct processData *)p;
    printf("\n\n\t\tBest fit");

    printf("\nProcess id\tProcess Size\tBlock No.\n");
    for (int i = 0; i < m; i++)
    {
        int x = -1;
        for (int j = 0; j < n; j++)
        {

            if (pd->space[j] >= pd->process_size[i])
            {
                if (x == -1)
                    x = j;
                else if (pd->space[x] > pd->space[j])
                    x = j;
            }
        }
        if (x != -1)
        {
            pd->allocation[i] = x;
            pd->space[x] -= pd->process_size[i];
        }
        printf("%d\t\t%d\t\t", i + 1, pd->process_size[i]);
        if (pd->allocation[i] == -1)
            printf("Not allocated\n");
        else
            printf("%i\n", pd->allocation[i] + 1);
    }
}
void *worstFit(void *p)
{
    struct processData *pd = (struct processData *)p;
    printf("\n\n\t\tWorst fit");

    printf("\nProcess id\tProcess Size\tBlock No.\n");
    for (int i = 0; i < m; i++)
    {
```

```
        int x = -1;
        for (int j = 0; j < n; j++)
        {

            if (pd->space[j] >= pd->process_size[i])
            {
                if (x == -1)
                    x = j;
                else if (pd->space[x] < pd->space[j])
                    x = j;
            }
        }
        if (x != -1)
        {
            pd->allocation[i] = x;
            pd->space[x] -= pd->process_size[i];
        }
        printf("%d\t\t%d\t\t", i + 1, pd->process_size[i]);
        if (pd->allocation[i] == -1)
            printf("Not allocated\n");
        else
            printf("%i\n", pd->allocation[i] + 1);
    }
}
int main()
{
    struct processData pd1, pd2, pd3;
    for (int i = 0; i < n; i++)
    {
        printf("\nEnter the divison number %d (in GB):", i + 1);
        scanf("%d", &pd1.space[i]);
    }
    for (int i = 0; i < m; i++)
    {
        printf("\nEnter the allocation required for process number %d (in GB):", i + 1);
        scanf("%d", &pd1.process_size[i]);
    }
    for (int i = 0; i < m; i++)
        pd1.allocation[i] = -1;
    pd2 = pd1;
    pd3 = pd1;
    pthread_t first_t, best_t, worst_t;
    pthread_create(&first_t, NULL, (void *)firstFit, (void *)&pd1);
    pthread_join(first_t, NULL);
    int flag = m;
    for (int i = 0; i < m; i++)
    {
        if (pd1.allocation[i] == -1)
        {
            printf("\nProcess Id %d is unallocated\n", i);
            flag--;
        }
    }
    if (m == flag)
```

```
        printf("\nAll process are successfully allocated\n");
    pthread_create(&best_t, NULL, (void *)bestFit, (void *)&pd2);
    pthread_join(best_t, NULL);
    flag = m;
    for (int i = 0; i < m; i++)
    {
        if (pd2.allocation[i] == -1)
        {
            printf("\nProcess Id %d is unallocated\n", i);
            flag--;
        }
    }
    if (m == flag)
        printf("\nAll process are successfully allocated\n");
    pthread_create(&worst_t, NULL, (void *)worstFit, (void *)&pd3);
    pthread_join(worst_t, NULL);
    flag = m;
    for (int i = 0; i < m; i++)
    {
        if (pd3.allocation[i] == -1)
        {
            printf("\nProcess Id %d is unallocated\n", i);
            flag--;
        }
    }
    if (m == flag)
        printf("\nAll process are successfully allocated\n");
    return 0;
}
```

Output:

```
Enter the divison number 1 (in GB):100

Enter the divison number 2 (in GB):500

Enter the divison number 3 (in GB):200

Enter the divison number 4 (in GB):300

Enter the divison number 5 (in GB):600

Enter the allocation required for process number 1 (in GB):212

Enter the allocation required for process number 2 (in GB):417

Enter the allocation required for process number 3 (in GB):112

Enter the allocation required for process number 4 (in GB):426
```

```
                First fit
Process id      Process Size    Block No.
1               212             2
2               417             5
3               112             2
4               426             Not allocated

Process Id 3 is unallocated


                Best fit
Process id      Process Size    Block No.
1               212             4
2               417             2
3               112             3
4               426             5

All process are successfully allocated


                Worst fit
Process id      Process Size    Block No.
1               212             5
2               417             2
3               112             5
4               426             Not allocated

Process Id 3 is unallocated
PS D:\VIT\Sem 2-2\OPERATING SYSTEMS\Assignments\lab 12>
```

3. Develop a C program to do best fit, worst fit, first fit memory allocation of fixed partition with assumption of block sizes and processes memory request sizes are in process.txt file. So your program should read the data from the file and perform the memory allocations.

Code:

```c
#include <stdio.h>
void firstFit(int process_size[], int m, int space[], int n)
{
    printf("\n\n\t\tFirst fit");
    int allocation[m];
    for (int i = 0; i < m; i++)
        allocation[i] = -1;
    printf("\nProcess id\tProcess Size\tBlock No.\n");
    for (int i = 0; i < m; i++)
    {
        for (int j = 0; j < n; j++)
        {
            if (space[j] >= process_size[i])
            {
                space[j] -= process_size[i];
                allocation[i] = j;
                break;
            }
```

```
        }
        printf("%d\t\t%d\t\t", i + 1, process_size[i]);
        if (allocation[i] == -1)
            printf("Not allocated\n");
        else
            printf("%i\n", allocation[i] + 1);
    }
}
void bestFit(int process_size[], int m, int space[], int n)
{
    printf("\n\n\t\tBest fit");
    int allocation[m];
    for (int i = 0; i < m; i++)
        allocation[i] = -1;
    printf("\nProcess id\tProcess Size\tBlock No.\n");
    for (int i = 0; i < m; i++)
    {
        int x = -1;
        for (int j = 0; j < n; j++)
        {

            if (space[j] >= process_size[i])
            {
                if (x == -1)
                    x = j;
                else if (space[x] > space[j])
                    x = j;
            }
        }
        if (x != -1)
        {
            allocation[i] = x;
            space[x] -= process_size[i];
        }
        printf("%d\t\t%d\t\t", i + 1, process_size[i]);
        if (allocation[i] == -1)
            printf("Not allocated\n");
        else
            printf("%i\n", allocation[i] + 1);
    }
}
void worstFit(int process_size[], int m, int space[], int n)
{
    printf("\n\n\t\tWorst fit");
    int allocation[m];
    for (int i = 0; i < m; i++)
        allocation[i] = -1;
    printf("\nProcess id\tProcess Size\tBlock No.\n");
    for (int i = 0; i < m; i++)
    {
        int x = -1;
        for (int j = 0; j < n; j++)
        {
```

```
            if (space[j] >= process_size[i])
            {
                if (x == -1)
                    x = j;
                else if (space[x] < space[j])
                    x = j;
            }
        }
        if (x != -1)
        {
            allocation[i] = x;
            space[x] -= process_size[i];
        }
        printf("%d\t\t%d\t\t", i + 1, process_size[i]);
        if (allocation[i] == -1)
            printf("Not allocated\n");
        else
            printf("%i\n", allocation[i] + 1);
    }
}
int main()
{
    int n, m;
    FILE *fp = fopen("process.txt", "r");
    fscanf(fp, "%d", &n);
    fscanf(fp, "%d", &m);
    int space1[n];
    int space2[n];
    int space3[n];
    for (int i = 0; i < n; i++)
    {
        fscanf(fp, "%d", &space1[i]);
        space2[i] = space1[i];
        space3[i] = space1[i];
    }
    int process_size[m];
    for (int i = 0; i < m; i++)
    {
        fscanf(fp, "%d", &process_size[i]);
    }

    firstFit(process_size, m, space1, n);
    bestFit(process_size, m, space2, n);
    worstFit(process_size, m, space3, n);
    fclose(fp);
    return 0;
}
```

```
PS D:\VIT\Sem 2-2\OPERATING SYSTEMS\Assignments\lab 12> cat process.txt
5
4
100
500
200
300
600
212
417
112
426
```

```
                First fit
Process id      Process Size    Block No.
1               212             2
2               417             5
3               112             2
4               426             Not allocated


                Best fit
Process id      Process Size    Block No.
1               212             4
2               417             2
3               112             3
4               426             5


                Worst fit
Process id      Process Size    Block No.
1               212             5
2               417             2
3               112             5
4               426             Not allocated
PS D:\VIT\Sem 2-2\OPERATING SYSTEMS\Assignments\lab 12>
```