Experiment no: 9

Name: Saloni Vishwakarma
Batch-Roll no: C1-13
Semester-Section: 4th-C
Subject: OS(Operating System) Lab
Date of execution: 12 July 2023

Aim: Write C programs to implement the following:
A) Disk scheduling algorithms
B) Page replacement algorithms and
C) Demonstrate different memory management schemes

Theory:

SSTF(Shortest Seek Time First): SSTF is a secondary storage scheduling algorithm that determines the motion of the disk's head and arm in servicing the read and write requests. SSTF acts as a disk scheduling algorithm, and it is an improvement upon the FCFS algorithm. It selects the requests with the least seek time starting from the current head position. Scheduling priority goes to the processes with the shortest seek- even if the requests aren't the first ones in the queue.

SSTF implements this by calculating the seek time of every request in the queue in advance. It then schedules the requests according to their seek time. However, this process doesn't ensure fairness and can lead to unspecified postponement. The seek pattern is highly localized in SSTF, and it is like SJF (Shortest Job First) because it can prevent distant requests from being serviced under heavy load. This phenomenon is known as Starvation.

FIFO(First In First Out): It is one of the types of page replacement algorithm. The FIFO algorithm is used in the paging method for memory management in an operating system that decides which existing page needs to be replaced in the queue. FIFO algorithm replaces the oldest (First) page which has been present for the longest time in the main memory.

In simple words, When a new page comes in from secondary memory to main memory, It selects the front of the queue which is the oldest page present, and removes it.

Worst Fit Memory allocation: In this allocation technique, the process traverses the whole memory and always search for the largest hole/partition, and then the process is placed in that hole/partition. It is a slow process because it has to traverse the entire memory to search the largest hole.

The algorithms searches sequentially starting from first memory block and searches for the memory block that fulfills the following condition –
-Can accommodate the process size
-Leaves the largest wasted space (fragmentation) after the process is allocated to given memory block

Viva Questions:

1) Why disk scheduling algorithms are important?
→Disk scheduling is done by operating systems to schedule I/O requests arriving for the disk. Disk scheduling is also known as I/O scheduling. Disk scheduling is important because:
• Multiple I/O requests may arrive by different processes and only one I/O request can be served at a time by the disk controller. Thus other I/O requests need to wait in the waiting queue and need to be scheduled.
• Two or more requests may be far from each other so can result in greater disk arm movement.
• Hard drives are one of the slowest parts of the computer system and thus need to be accessed in an efficient manner

2) What are the advantages of SSTF algorithm- Shortest Seek Time First?
• Better performance than FCFS scheduling algorithm
• It provides better throughput.
• This algorithm is used in Batch Processing system where throughput is more important.
• It has less average response and waiting time.

3) What is Page Replacement Algorithm?
→Page replacement algorithms are an important part of virtual memory management and it helps the OS to decide which memory page can be moved out, making space for the currently needed page.

4) What are the advantages of FIFO?
• It is simple and easy to understand & implement.
• It is efficiently used for small systems
• It does not cause more overheads

5) Why memory management is required?

• Allocate and de-allocate memory before and after process execution.
• To keep track of used memory space by processes.
• To minimize fragmentation issues.
• To proper utilization of main memory.
• To maintain data integrity while executing of process.

6) What are the advantages and disadvantages of worst fit allocation?

→Advantages of Worst-Fit Allocation : Since this process chooses the largest hole/partition, therefore there will be large internal fragmentation. Now, this internal fragmentation will be quite big so that other small processes can also be placed in that leftover partition.

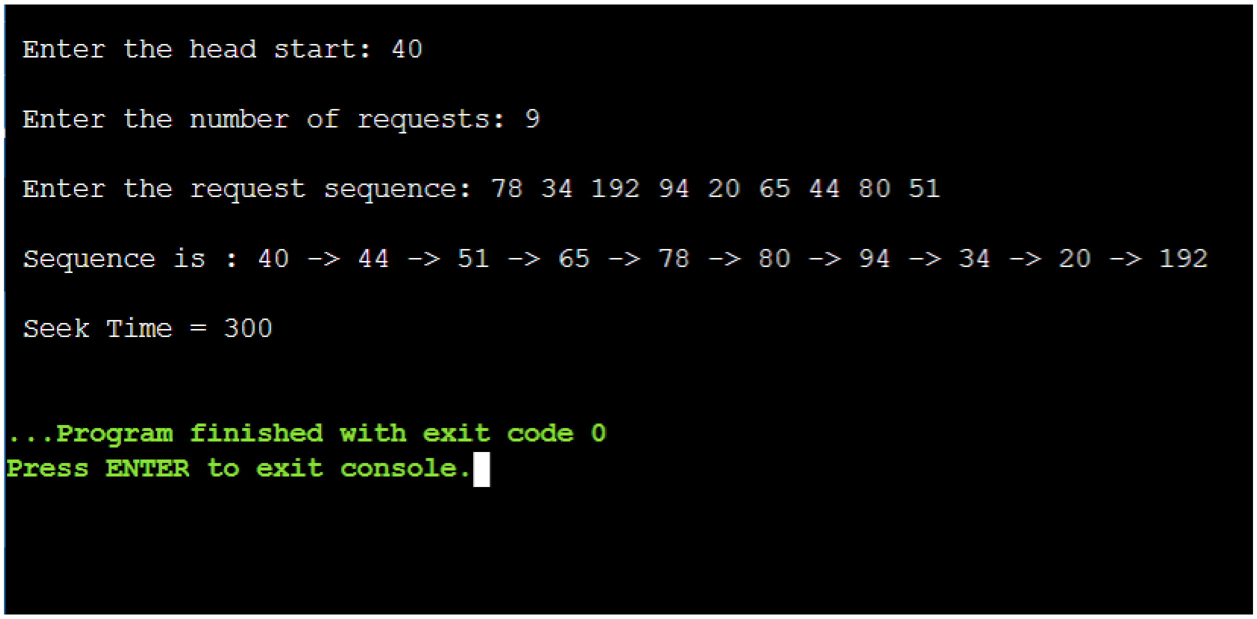
Disadvantages of Worst-Fit Allocation : It is a slow process because it traverses all the partitions in the memory and then selects the largest partition among all the partitions, which is a time-consuming process.

Code and Output:

A) SSTF (Shortest Seek Time First):

```
#include<math.h>
#include<stdio.h>
#include<stdlib.h>
int main()
{
   int i,n,k,req[50],mov=0,cp,index[50],min,a[50],j=0,mini,cp1;
   printf("\n Enter the head start: ");
   scanf("%d",&cp);
   printf("\n Enter the number of requests: ");
   scanf("%d",&n);
   cp1=cp;
   printf("\n Enter the request sequence: ");
   for(i=0;i<n;i++)
   {
      scanf("%d",&req[i]);
   }
   for(k=0;k<n;k++)
   {
      for(i=0;i<n;i++)
      {
         index[i]=abs(cp-req[i]); // calculate distance of each request from current position
```

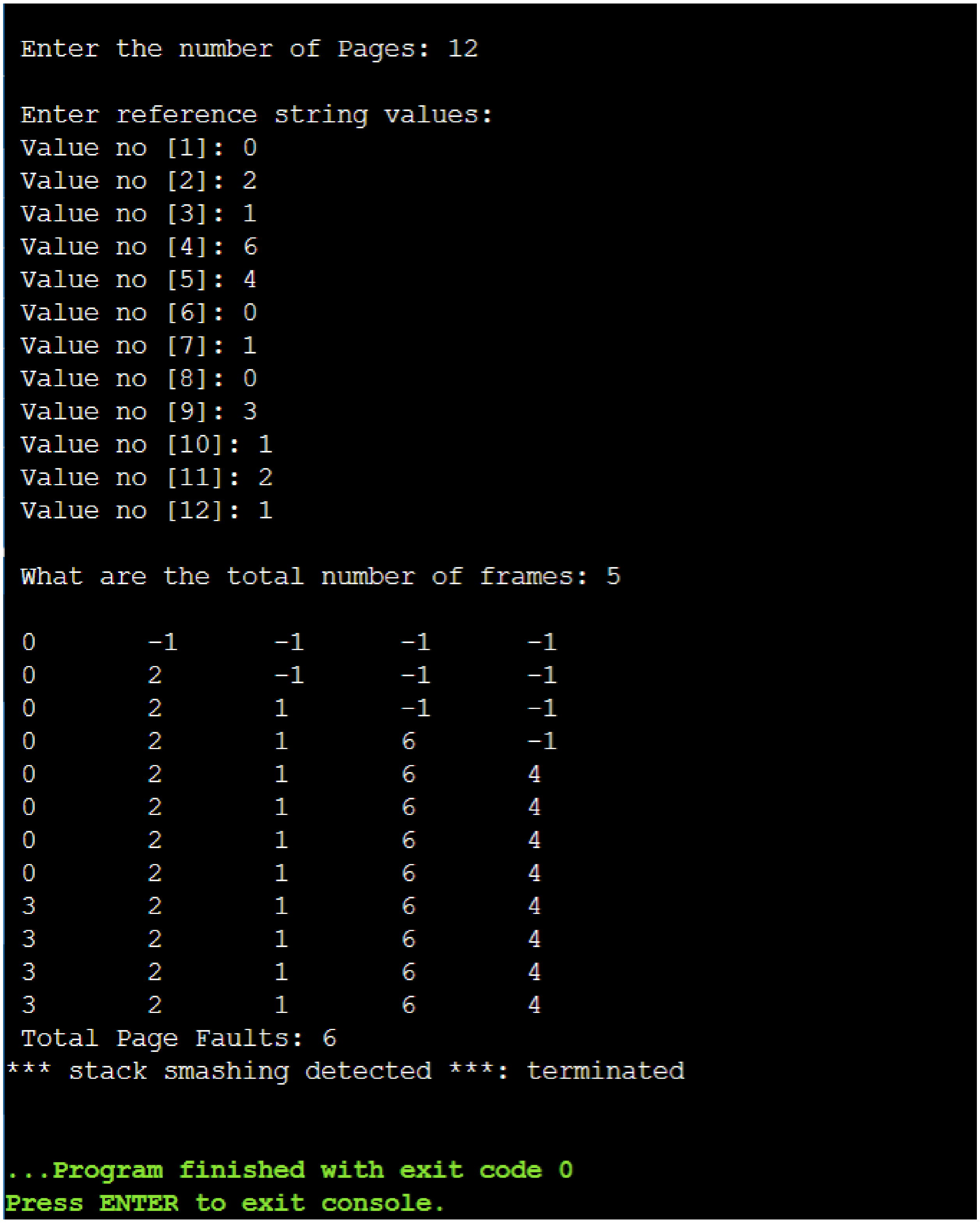
```
        }
        // to find the nearest request
        min=index[0];
        mini=0;
        for(i=1;i<n;i++)
        {
            if(min>index[i])
            {
                min=index[i];
                mini=i;
            }
        }
        a[j]=req[mini];
        j++;
        cp=req[mini]; // change the current position value to next request
        req[mini]=999;
    }
    // the request that is processed its value is changed so that it is not processed again
    printf("\n Sequence is : ");
    printf("%d",cp1);
    mov=mov+abs(cp1-a[0]); // head movement
    printf(" -> %d",a[0]);
    for(i=1;i<n;i++)
    {
        mov=mov+abs(a[i]-a[i-1]); ///head movement
        printf(" -> %d",a[i]);
    }
    printf("\n");
    printf("\n Seek Time = %d\n",mov);
}
```



```
Enter the head start: 40

Enter the number of requests: 9

Enter the request sequence: 78 34 192 94 20 65 44 80 51

Sequence is : 40 -> 44 -> 51 -> 65 -> 78 -> 80 -> 94 -> 34 -> 20 -> 192

Seek Time = 300


...Program finished with exit code 0
Press ENTER to exit console.
```

B) FIFO (First In First Out):

```
#include <stdio.h>
int main()
{
   int referenceString[10], pageFaults = 0, m, n, s, pages, frames;
   printf("\n Enter the number of Pages: ");
   scanf("%d", &pages);
   printf("\n Enter reference string values:\n");
   for( m = 0; m < pages; m++)
   {
      printf(" Value no [%d]: ", m + 1);
      scanf("%d", &referenceString[m]);
   }
   printf("\n What are the total number of frames: ");
   {
      scanf("%d", &frames);
   }
   int temp[frames];
   for(m = 0; m < frames; m++)
   {
      temp[m] = -1;
   }
   for(m = 0; m < pages; m++)
   {
        s = 0;
        for(n = 0; n < frames; n++)
        {
            if(referenceString[m] == temp[n])
            {
                s++;
                pageFaults--;
            }
        }
        pageFaults++;
        if((pageFaults <= frames) && (s == 0))
        {
            temp[m] = referenceString[m];
        }
        else if(s == 0)
        {
            temp[(pageFaults - 1) % frames] = referenceString[m];
        }
        printf("\n");
        for(n = 0; n < frames; n++)
        {
            printf(" %d\t", temp[n]);
        }
    }
    printf("\n Total Page Faults: %d\n", pageFaults);
    return 0;
}
```

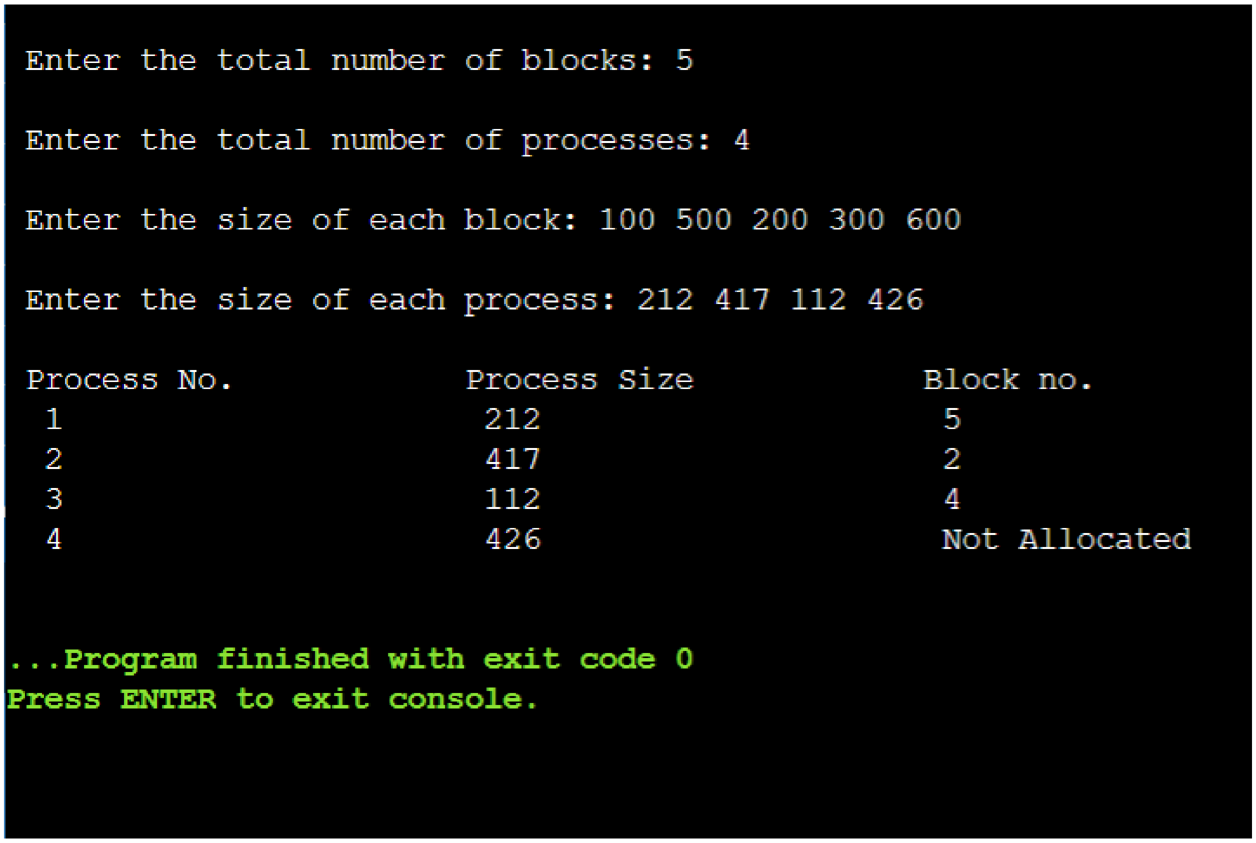


```
 Enter the number of Pages: 12

 Enter reference string values:
 Value no [1]: 0
 Value no [2]: 2
 Value no [3]: 1
 Value no [4]: 6
 Value no [5]: 4
 Value no [6]: 0
 Value no [7]: 1
 Value no [8]: 0
 Value no [9]: 3
 Value no [10]: 1
 Value no [11]: 2
 Value no [12]: 1

 What are the total number of frames: 5

 0       -1      -1      -1      -1
 0       2       -1      -1      -1
 0       2       1       -1      -1
 0       2       1       6       -1
 0       2       1       6       4
 0       2       1       6       4
 0       2       1       6       4
 0       2       1       6       4
 3       2       1       6       4
 3       2       1       6       4
 3       2       1       6       4
 3       2       1       6       4
 Total Page Faults: 6
*** stack smashing detected ***: terminated


...Program finished with exit code 0
Press ENTER to exit console.
```

C) Worst Fit :

```
#include <stdio.h>
void implementWorstFit(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("\n Process No.\t\tProcess Size\t\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 blocks,processes,i,j;
    printf("\n Enter the total number of blocks: ");
    scanf("%d",&blocks);
    printf("\n Enter the total number of processes: ");
    scanf("%d",&processes);
    int blockSize[blocks],processSize[processes];
    printf("\n Enter the size of each block: ");
    for(int i=0;i<blocks;i++)
    {
        scanf("%d",&blockSize[i]);
    }
    printf("\n Enter the size of each process: ");
    for(int i=0;i<processes;i++)
    {
        scanf("%d",&processSize[i]);
    }
    implementWorstFit(blockSize, blocks, processSize, processes);
    return 0;
}
```



```
Enter the total number of blocks: 5

Enter the total number of processes: 4

Enter the size of each block: 100 500 200 300 600

Enter the size of each process: 212 417 112 426

Process No.            Process Size            Block no.
 1                      212                     5
 2                      417                     2
 3                      112                     4
 4                      426                     Not Allocated


...Program finished with exit code 0
Press ENTER to exit console.
```

Conclusion: We have successfully studied and implemented SSTF-disk scheduling algorithm, FIFO-page replacement algorithm and Worst fit memory allocation using C.