SSN COLLEGE OF ENGINEERING, KALAVAKKAM
(An Autonomous Institution, Affiliated to Anna University, Chennai)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

UCS1411 - OPERATING SYSTEMS LAB

LAB EXERCISE 9

Paging Technique

Submission Date:23-05-2022

Name: Jayannthan P T *Dept: CSE 'A'* *Roll No.: 205001049*

1. To develop a C program to implement the paging technique in memory management

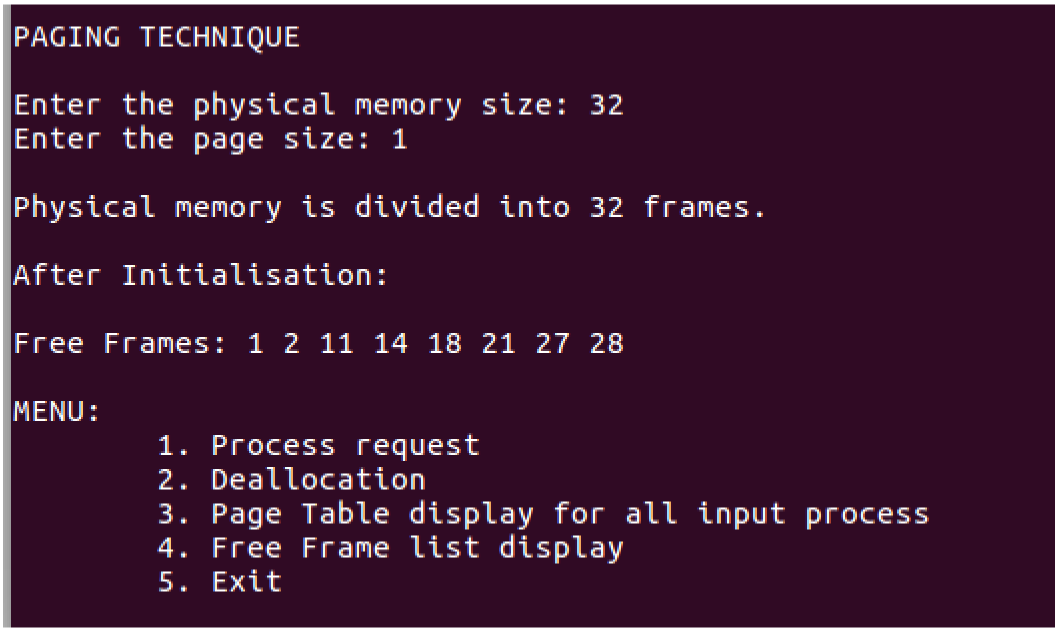
Algorithm:

1. Generate a random number using the srand() function.
2. Create an array of processes of type table.
3. Input the size of the physical memory and the size of each page.
4. Assign the ceil value of the quotient when the size of physical memory is divided by size of page as number of frames.
5. Initially, assign 0 to all elements of the frames array.
6. Assign -1 to random frames.
7.-1 indicate that the frames are free
8. Input an integer option.
9. If option is equal to 1, execute steps 9.1 to 9.6.
    9.1. Input the process ID and its size.
    9.2. Calculate the number of frames that will be occupied by the process.
    9.3. If the number of frames required by the process is less than the number of free frames allocate. Else, print that there aren't enough frames.
10. If option is equal to 2, execute steps 10.1 to 10.2.
    10.1. Input the ID of the process to be deallocated.
    10.2. Assign -1 to all the frames allocated to the process to indicate that the process frames have been deallocated. Remove the process from the process list as well.
11. If option is equal to 3, print the page table for all processes.
12. If option is equal to 4, print all the free frames, i.e., print all the frames where the (j+1)th element in the frames array is equal to -1.
13. If option is equal to 5, exit the program.
14. Else, declare that the user has entered an invalid input.

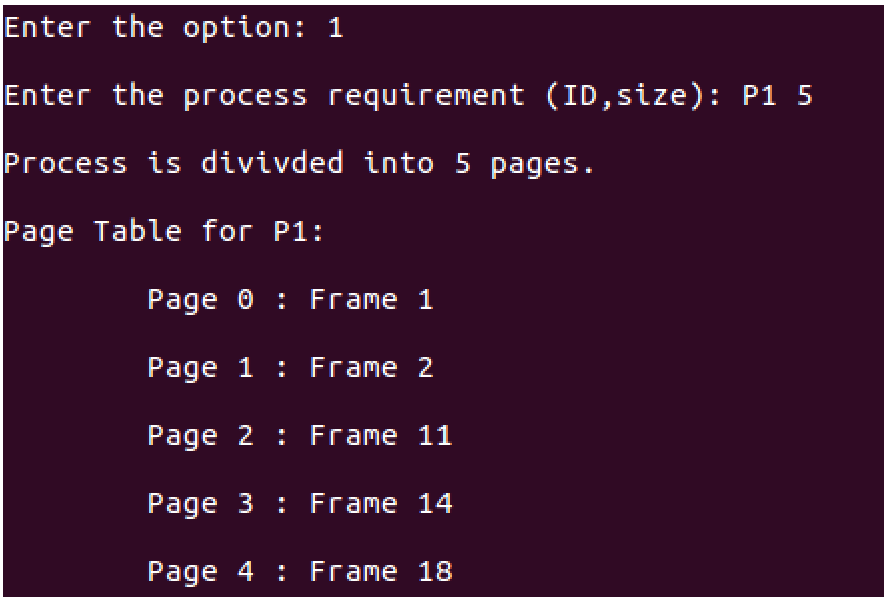
Code:
```c
#include <stdio.h>

#include <stdlib.h>
#include <math.h>
#include <time.h>
```

```
#define MAX 50
struct table
{
    int page[MAX];
    int frame[MAX];
    int no_of_pages;
};
int main()
{
    srand(time(0));
    int no_of_frames, option, process, frames_req, no_of_free_frames = 0, no_of_process =
0, m = 1;
    double size, memory_size, page_size;
    int frames[MAX], process_[MAX];
    struct table processes_[MAX];
    printf("\nPAGING TECHNIQUE\n");
    printf("\nEnter the physical memory size: ");
    scanf(" %lf", &memory_size);
    printf("Enter the page size: ");
    scanf(" %lf", &page_size);
    no_of_frames = ceil(memory_size / page_size);
    printf("\nPhysical memory is divided into %d frames.", no_of_frames);
    for (int i = 1; i <= no_of_frames; i++)
        frames[i] = 0;
    int n = 10;
    for (int i = 1; i <= n; i++)
    {
        frames[rand() % no_of_frames + 1] = -1;
        no_of_free_frames++;
    }
    printf("\n\nAfter Initialisation: \n\nFree Frames: ");
    for (int i = 1; i <= no_of_frames; i++)
    {
        if (frames[i] == -1)
            printf("%d ", i);
    }
    int p = 0;
    printf("\n\nMENU:\n\t1. Process request\n\t2. Deallocation\n\t3. Page Table display
for all input process\n\t4. Free Frame list display\n\t5. Exit\n");
    do
    {
        printf("\nEnter the option: ");
        scanf("%d", &option);
        switch (option)
        {
        case 1:
            printf("\nEnter the process requirement (ID,size): P");
            scanf(" %d %lf", &process, &size);
            frames_req = ceil(size / page_size);
            processes_[process].no_of_pages = frames_req;
            printf("\nProcess is divivded into %d pages.\n\nPage Table for P%d:\n",
frames_req, process);
            if (frames_req <= no_of_free_frames)
            {
                p = 0;
                no_of_process++;
                process_[no_of_process] = process;
                for (int i = 1; i <= frames_req;)
                {
                    for (int j = m, k = 1; k <= no_of_frames; k++)
                    {
                        if (frames[j] == -1)
                        {
                            printf("\n\tPage %d : Frame %d \n", i - 1, j);
                            no_of_free_frames--;
                            frames[j] = process;
                            processes_[process].page[p] = i - 1;
                            processes_[process].frame[p] = j;
                            i++;
                            p++;
                            break;
                        }
                        j = j % no_of_frames + 1;
                        m = j;
                    }
                }
            }
            else
                printf("\nThere is no enough free frames to allocate for this
process!\n");
            break;
        case 2:
            printf("\nEnter the process to be deallocated: P");
            scanf(" %d", &process);
            for (int i = 1; i <= no_of_frames; i++)
            {
                if (frames[i] == process)
                {
                    frames[i] = -1;
                    no_of_free_frames++;
                }
            }
            for (int i = 1; i <= no_of_process; i++)
            {
                if (process_[i] == process)
                {
                    process_[i] = -1;
                    break;
                }
            }
            break;
        case 3:
            for (int k = 1; k <= no_of_process; k++)
            {
                if (process_[k] != -1)
                {
                    printf("\nPage table for P%d:\n", process_[k]);
                    int i = 0;
                    for (; i < processes_[process_[k]].no_of_pages; i++)
                        printf("\n\tPage %d : Frame %d \n",
processes_[process_[k]].page[i], processes_[process_[k]].frame[i]);
                }
            }
            break;
        case 4:
            printf("\nFree Frames: ");
            for (int i = 1, j = m; i <= no_of_frames; i++)
            {
                if (frames[j] == -1)
                    printf("%d ", j);
                j = j % no_of_frames + 1;
                m = j;
            }
            printf("\n");
            break;
        case 5:
            printf("\nProgram terminated\n");
            break;
        default:
            printf("\nInvalid option\n");
            break;
        }
    } while (option != 5);
}
```

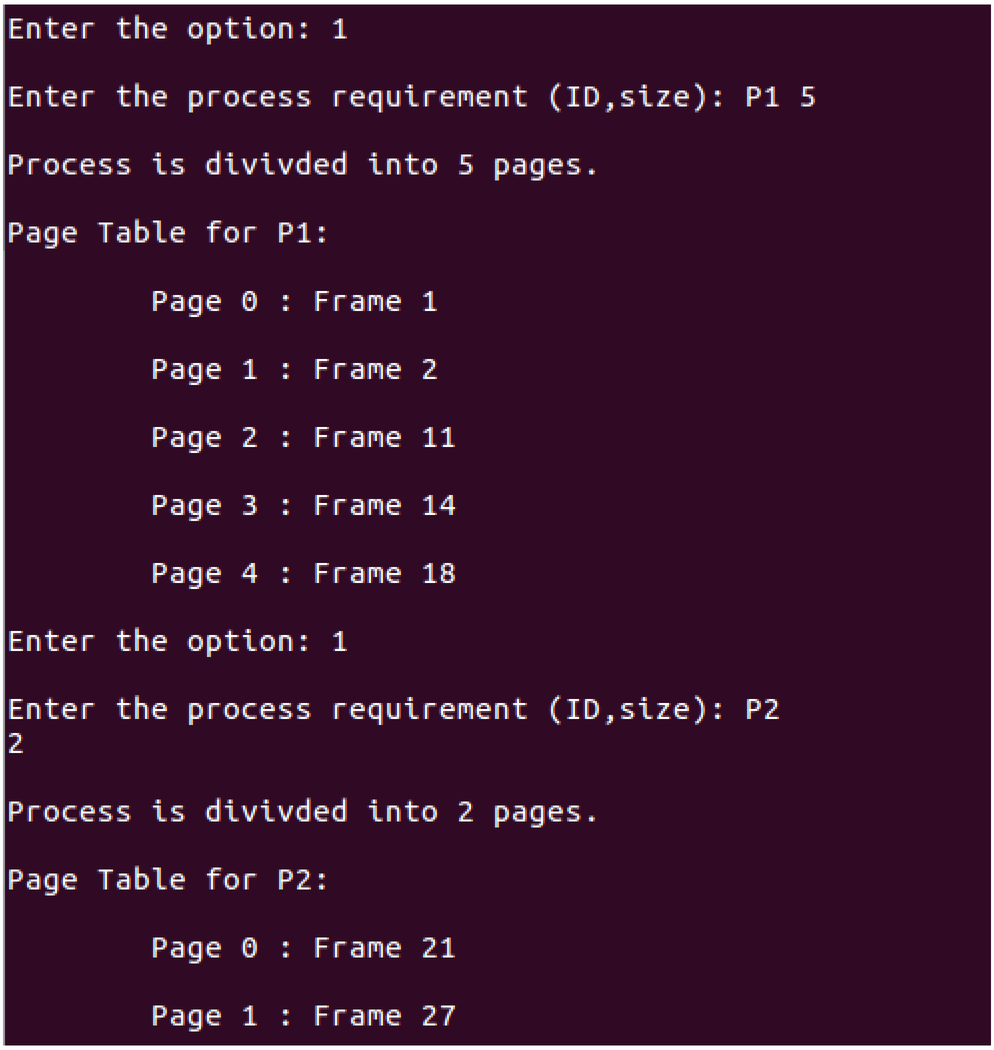
Output:



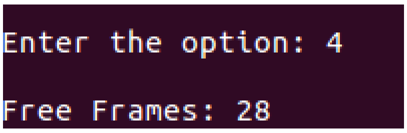
```
PAGING TECHNIQUE

Enter the physical memory size: 32
Enter the page size: 1

Physical memory is divided into 32 frames.

After Initialisation:

Free Frames: 1 2 11 14 18 21 27 28

MENU:
        1. Process request
        2. Deallocation
        3. Page Table display for all input process
        4. Free Frame list display
        5. Exit
```



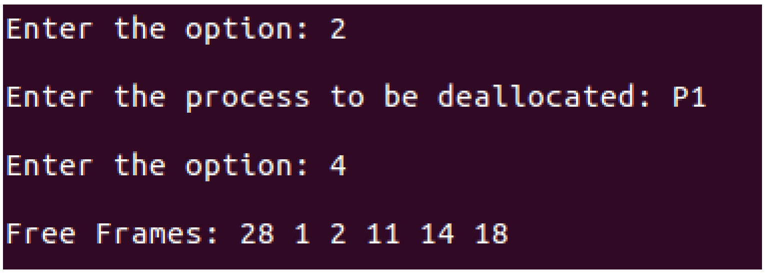
```
Enter the option: 1

Enter the process requirement (ID,size): P1 5

Process is divivded into 5 pages.

Page Table for P1:

        Page 0 : Frame 1

        Page 1 : Frame 2

        Page 2 : Frame 11

        Page 3 : Frame 14

        Page 4 : Frame 18
```



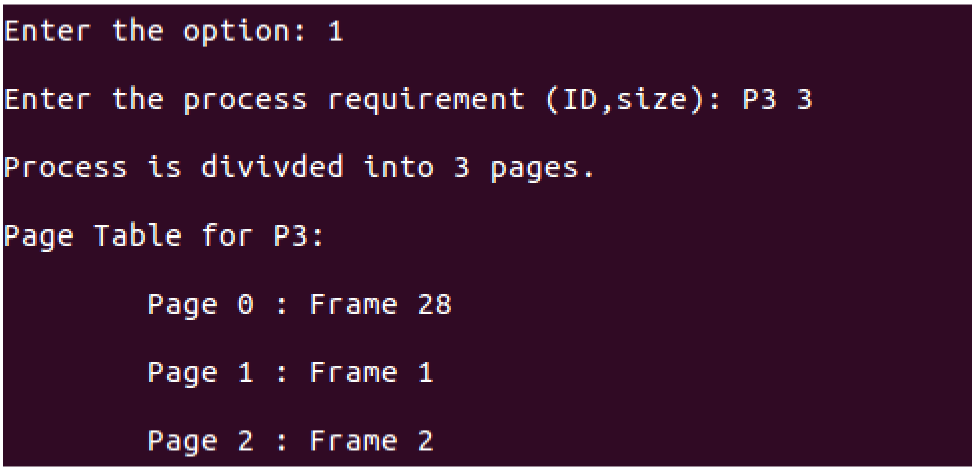
```
Enter the option: 1

Enter the process requirement (ID,size): P1 5

Process is divivded into 5 pages.

Page Table for P1:

        Page 0 : Frame 1

        Page 1 : Frame 2

        Page 2 : Frame 11

        Page 3 : Frame 14

        Page 4 : Frame 18

Enter the option: 1

Enter the process requirement (ID,size): P2
2

Process is divivded into 2 pages.

Page Table for P2:

        Page 0 : Frame 21

        Page 1 : Frame 27
```



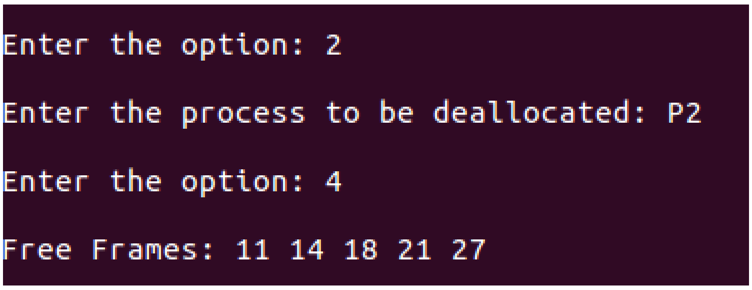
```
Enter the option: 4

Free Frames: 28
```



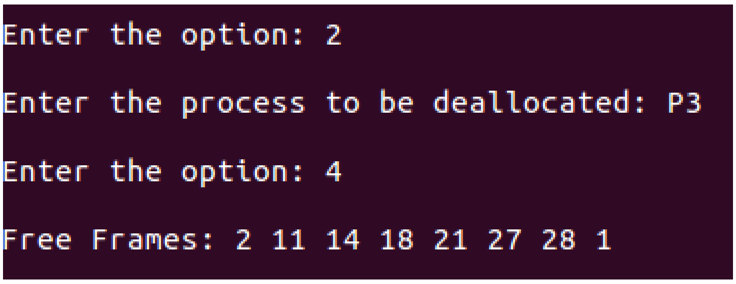
```
Enter the option: 2

Enter the process to be deallocated: P1

Enter the option: 4

Free Frames: 28 1 2 11 14 18
```



```
Enter the option: 1

Enter the process requirement (ID,size): P3 3

Process is divivded into 3 pages.

Page Table for P3:

        Page 0 : Frame 28

        Page 1 : Frame 1

        Page 2 : Frame 2
```



```
Enter the option: 2

Enter the process to be deallocated: P2

Enter the option: 4

Free Frames: 11 14 18 21 27
```



```
Enter the option: 2

Enter the process to be deallocated: P3

Enter the option: 4

Free Frames: 2 11 14 18 21 27 28 1
```

Learning Outcome:

- Learned about the use of paging technique.
- Learned how to allocate memory to processes using paging technique.