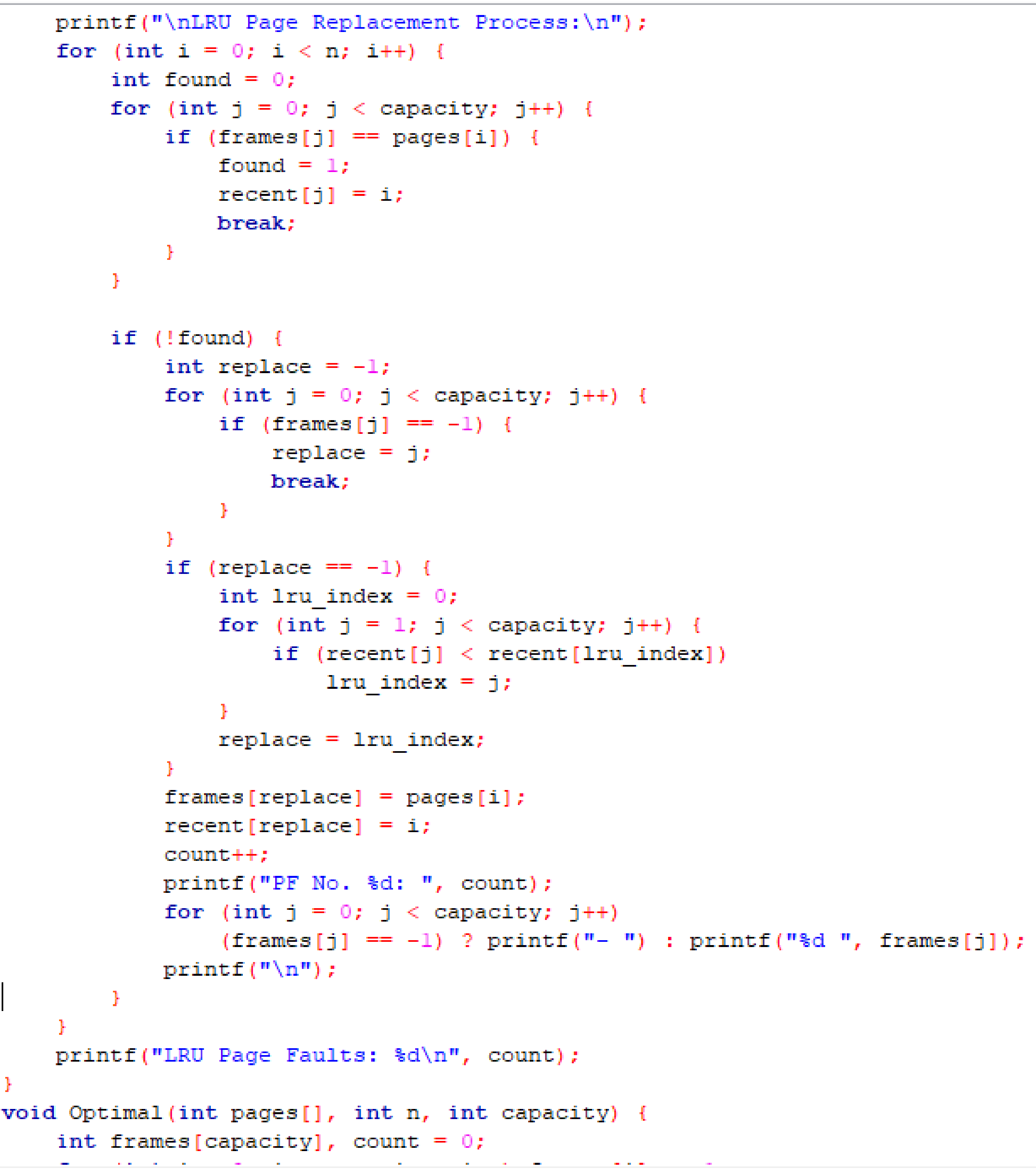
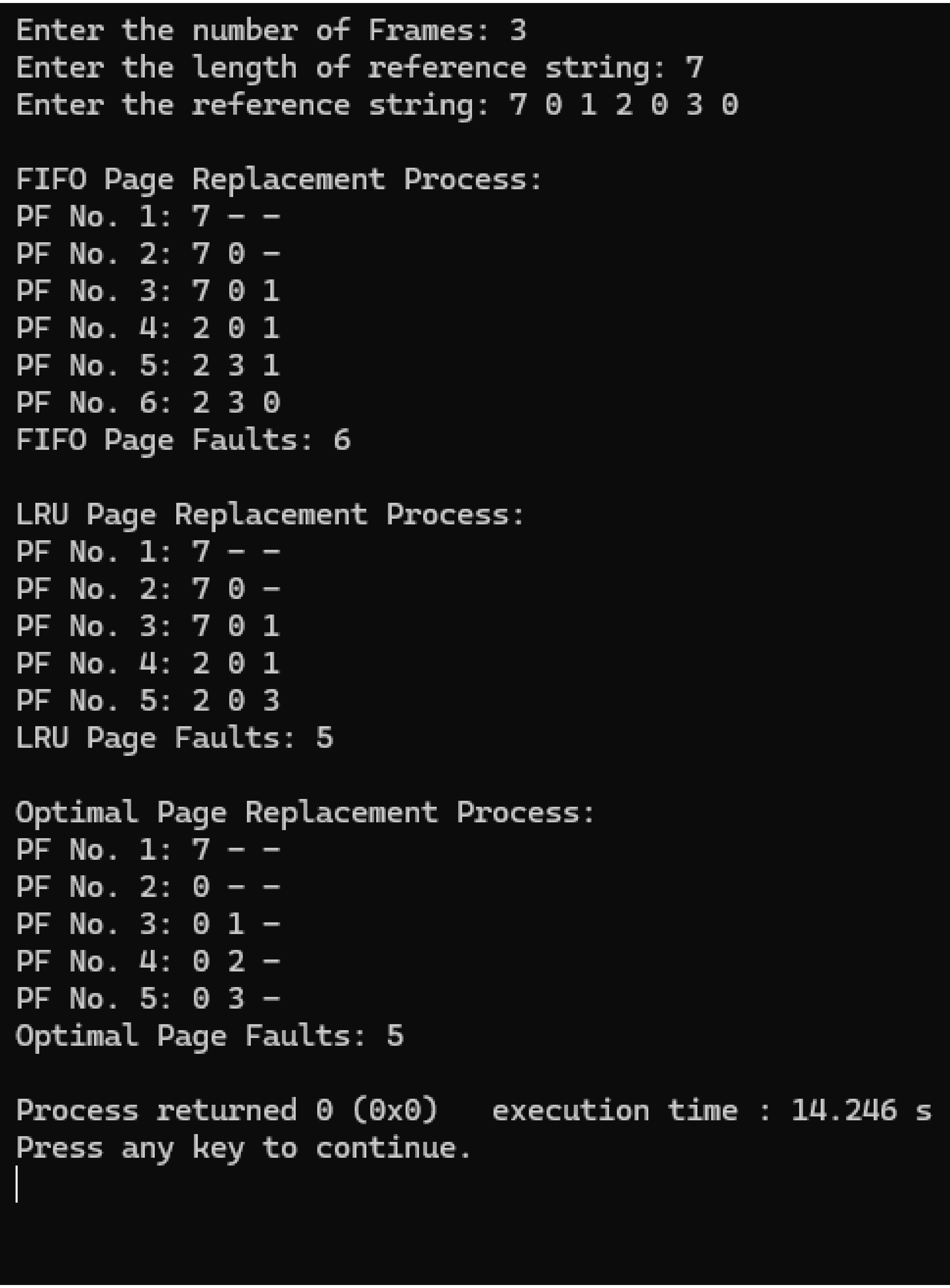
```c
#include <stdio.h>
#define MAX 100
int isPresent(int frames[], int page, int n) {
    for (int i = 0; i < n; i++)
        if (frames[i] == page)
            return 1;
    return 0;
}
void FIFO(int pages[], int n, int capacity) {
    int frames[capacity], front = 0, count = 0;
    for (int i = 0; i < capacity; i++) frames[i] = -1;

    printf("\nFIFO Page Replacement Process:\n");
    for (int i = 0; i < n; i++) {
        if (!isPresent(frames, pages[i], capacity)) {
            frames[front] = pages[i];
            front = (front + 1) % capacity;
            count++;
            printf("PF No. %d: ", count);
            for (int j = 0; j < capacity; j++)
                (frames[j] == -1) ? printf("- ") : printf("%d ", frames[j]);
            printf("\n");
        }
    }
    printf("FIFO Page Faults: %d\n", count);
}
void LRU(int pages[], int n, int capacity) {
    int frames[capacity], recent[capacity], count = 0;
    for (int i = 0; i < capacity; i++) {
        frames[i] = -1;
        recent[i] = -1;
    }

    printf("\nLRU Page Replacement Process:\n");
    for (int i = 0; i < n; i++) {
        int found = 0;
        for (int j = 0; j < capacity; j++) {
            if (frames[j] == pages[i]) {
                found = 1;
                recent[j] = i;
```



```c
    printf("\nLRU Page Replacement Process:\n");
    for (int i = 0; i < n; i++) {
        int found = 0;
        for (int j = 0; j < capacity; j++) {
            if (frames[j] == pages[i]) {
                found = 1;
                recent[j] = i;
                break;
            }
        }

        if (!found) {
            int replace = -1;
            for (int j = 0; j < capacity; j++) {
                if (frames[j] == -1) {
                    replace = j;
                    break;
                }
            }
            if (replace == -1) {
                int lru_index = 0;
                for (int j = 1; j < capacity; j++) {
                    if (recent[j] < recent[lru_index])
                        lru_index = j;
                }
                replace = lru_index;
            }
            frames[replace] = pages[i];
            recent[replace] = i;
            count++;
            printf("PF No. %d: ", count);
            for (int j = 0; j < capacity; j++)
                (frames[j] == -1) ? printf("- ") : printf("%d ", frames[j]);
            printf("\n");
        }
    }
    printf("LRU Page Faults: %d\n", count);
}
void Optimal(int pages[], int n, int capacity) {
    int frames[capacity], count = 0;
    printf("LRU Page Faults: %d\n", count);
}
void Optimal(int pages[], int n, int capacity) {
    int frames[capacity], count = 0;
    for (int i = 0; i < capacity; i++) frames[i] = -1;

    printf("\nOptimal Page Replacement Process:\n");
    for (int i = 0; i < n; i++) {
        if (!isPresent(frames, pages[i], capacity)) {
            int index = -1, farthest = i;
            for (int j = 0; j < capacity; j++) {
                int k;
                for (k = i + 1; k < n; k++) {
                    if (frames[j] == pages[k]) break;
                }
                if (k > farthest) {
                    farthest = k;
                    index = j;
                }
            }

            if (index == -1) {
                for (int j = 0; j < capacity; j++) {
                    if (frames[j] == -1) {
                        index = j;
                        break;
                    }
                }
                if (index == -1)
                    index = 0;
            }

            frames[index] = pages[i];
            count++;
            printf("PF No. %d: ", count);
            for (int j = 0; j < capacity; j++)
                (frames[j] == -1) ? printf("- ") : printf("%d ", frames[j]);
            printf("\n");
        }
    }
```

```c
                for (int j = 0; j < capacity; j++) {
                    if (frames[j] == -1) {
                        index = j;
                        break;
                    }
                }
                if (index == -1)
                    index = 0;
            }

            frames[index] = pages[i];
            count++;
            printf("PF No. %d: ", count);
            for (int j = 0; j < capacity; j++)
                (frames[j] == -1) ? printf("- ") : printf("%d ", frames[j]);
            printf("\n");
        }
    }
    printf("Optimal Page Faults: %d\n", count);
}
int main() {
    int capacity, n, pages[MAX];

    printf("Enter the number of Frames: ");
    scanf("%d", &capacity);

    printf("Enter the length of reference string: ");
    scanf("%d", &n);

    printf("Enter the reference string: ");
    for (int i = 0; i < n; i++)
        scanf("%d", &pages[i]);

    FIFO(pages, n, capacity);
    LRU(pages, n, capacity);
    Optimal(pages, n, capacity);

    return 0;
}
```



```
Enter the number of Frames: 3
Enter the length of reference string: 7
Enter the reference string: 7 0 1 2 0 3 0

FIFO Page Replacement Process:
PF No. 1: 7 - -
PF No. 2: 7 0 -
PF No. 3: 7 0 1
PF No. 4: 2 0 1
PF No. 5: 2 3 1
PF No. 6: 2 3 0
FIFO Page Faults: 6

LRU Page Replacement Process:
PF No. 1: 7 - -
PF No. 2: 7 0 -
PF No. 3: 7 0 1
PF No. 4: 2 0 1
PF No. 5: 2 0 3
LRU Page Faults: 5

Optimal Page Replacement Process:
PF No. 1: 7 - -
PF No. 2: 0 - -
PF No. 3: 0 1 -
PF No. 4: 0 2 -
PF No. 5: 0 3 -
Optimal Page Faults: 5

Process returned 0 (0x0)   execution time : 14.246 s
Press any key to continue.
```