PRACTICAL NO 6: PAGING [LRU METHOD]

```
#include <stdio.h>
#define INFINITE 1000
int search(int a[], int n, int pageno) {
 int i;
 for (i = 0; i < n; i++)
 if (a[i] == pageno)
 return 1;
 return 0;
}
int findmax(int a[], int n) {
 int i, j;
 j = 0;
 for (i = 1; i < n; i++)
 if (a[i] > a[j])
 j = i;
 return j;
}
int findempty(int a[], int n) {
 int i;
 for (i = 0; i < n; i++)
 if (a[i] == -1)
 return i;
 return -1;
}
int main() {
 int lruf[10], trace[30], ntrace, nframes;
 int i, j, loc, lrud[10];
 int page_faults = 0;
 printf("\nEnter no. of frames: ");
 scanf("%d", &nframes);
 printf("\nEnter no of entries in the page trace: ");
 scanf("%d", &ntrace);
 printf("\nEnter page trace: ");
 for (i = 0; i < ntrace; i++)
 scanf("%d", &trace[i]);
 for (i = 0; i < nframes; i++) {
 lruf[i] = -1;
 lrud[i] = 0;
 }
 printf("\nPage no. LRU Allocation");
 for (i = 0; i < ntrace; i++) {
 if (!search(lruf, nframes, trace[i])) {
 loc = findempty(lruf, nframes);
 if (loc != -1) { // Empty frame
 for (j = 0; j < nframes; j++)
 lrud[j]++;
 lruf[loc] = trace[i];
 lrud[loc] = 0;
 } else {
 loc = findmax(lrud, nframes);
 lruf[loc] = trace[i];
 for (j = 0; j < nframes; j++)
 lrud[j]++;
 lrud[loc] = 0;
 }
 page_faults++; // Increment page faults only when a new page is loaded
 } else {
 for (j = 0; j < nframes; j++) {
 if (lruf[j] != trace[i])
 lrud[j]++;
 else
 lrud[j] = 0;
```

```
 }
 }
 printf("\n %d ", trace[i]);
 for (j = 0; j < nframes; j++)
 printf("%3d ", lruf[j]);
 }
 printf("\nPAGE FAULTS: %d", page_faults);
 return 0;
}
```

OUTPUT:

```
Enter no. of frames: 3

Enter no of entries in the page trace: 7

Enter page trace: 1
2
3
4
1
2
5

Page no. LRU Allocation
 1       1  -1  -1
 2       1   2  -1
 3       1   2   3
 4       4   2   3
 1       4   1   3
 2       4   1   2
 5       5   1   2
PAGE FAULTS: 7
```