PROGRAM CODE

```
#include <stdio.h>
#include <stdlib.h>

struct pagetable {
	int num;
	int index;
};

struct pages {
	int num;
	int count;
};

void fifo(int m, int n, int pages[]) {
	printf("\nFIFO\n");

	struct pagetable table[m];
	int index = 0, free = 1, faults = 0;

	for (int i=0; i<n; i++) {
		printf("%d: ", pages[i]);

		int contains = 0;

		for (int j=0; j<m; j++)
			if (table[j].num == pages[i]) {
				contains = 1;
				break;
			}

		if (contains) {
			for (int j=0; j<m; j++) {
				if (free) {
					if (j < index)
						printf("%d ", table[j].num);
					else
						printf(" ");
				} else
					printf("%d ", table[j].num);
			}
			printf("\n");
		} else {
			table[index].num = pages[i];
			index = (index + 1) % m;
			faults++;

			if (index == 0)
				free = 0;

			for (int j=0; j<m; j++) {
				if (free) {
					if (j < index)
						printf("%d ", table[j].num);
					else
```

```
                                        printf(" ");
                                } else
                                        printf("%d ", table[j].num);
                        }
                        printf("\n");
                }
        }

        printf("\nNo of page faults = %d\n", faults);
}

void lru(int m, int n, int pages[]) {
        printf("\nLRU\n");

        struct pagetable table[m];
        int index = -1, free = 1, faults = 0, count = 0;

        for (int i=0; i<n; i++) {
                printf("%d: ", pages[i]);

                int contains = 0;

                for (int j=0; j<m; j++)
                        if (table[j].num == pages[i]) {
                                table[j].index = count;
                                count++;

                                for (int j=0; j<m; j++) {
                                        if (free) {
                                                if (j <= index)
                                                        printf("%d ", table[j].num);
                                                else
                                                        printf(" ");
                                        } else
                                                printf("%d ", table[j].num);
                                }
                                printf("\n");

                                contains = 1;
                        }

                if (contains == 0) {
                        if (free) {
                                index = (index + 1) % m;

                                if (index == (m-1))
                                        free = 0;
                        }
                        else {
                                index = 0;

                                for (int j=1; j<m; j++)
                                        if (table[j].index < table[index].index)
                                                index = j;
                        }

                        table[index].num = pages[i];
```

```
                        table[index].index = count;
                        count++;
                        faults++;

                        for (int j=0; j<m; j++) {
                                if (free) {
                                        if (j <= index)
                                                printf("%d ", table[j].num);
                                        else
                                                printf(" ");
                                } else
                                        printf("%d ", table[j].num);
                        }
                        printf("\n");
                }
        }

        printf("\nNo of page faults = %d\n", faults);
}

void lfu(int m, int n, int pages[]) {
        printf("\nLFU\n");

        struct pagetable table[m];
        struct pages map[n];
        int index = -1, free = 1, faults = 0, count = 0, maplen = 0;

        for (int i=0; i<n; i++) {
                printf("%d: ", pages[i]);

                int contains = 0;

                for (int j=0; j<m; j++)
                        if (table[j].num == pages[i]) {
                                for (int k=0; k<maplen; k++)
                                        if (map[k].num == table[j].num) {
                                                map[k].count++;
                                                break;
                                        }

                                table[j].index = count;
                                count++;

                                for (int j=0; j<m; j++) {
                                        if (free) {
                                                if (j <= index)
                                                        printf("%d ", table[j].num);
                                                else
                                                        printf(" ");
                                        } else
                                                printf("%d ", table[j].num);
                                }
                                printf("\n");

                                contains = 1;
                        }
```

```
if (contains == 0) {
	if (free) {
		index = (index + 1) % m;

		if (index == (m-1))
			free = 0;
	}
	else {
		index = 0;
		int index1 = 0, index2 = 0;

		for (int j=1; j<m; j++) {
			for (int k=0; k<maplen; k++)
				if (map[k].num == table[index].num) {
					index1 = k;
					continue;
				} else if (map[k].num == table[j].num) {
					index2 = k;
					continue;
				}

			if (map[index2].count < map[index1].count) {
				index = j;
			} else if (map[index2].count == map[index1].count) {
				if (table[j].index < table[index].index)
					index = j;
			}
		}
	}

	table[index].num = pages[i];

	int exists = 0;

	for (int k=0; k<maplen; k++)
		if (map[k].num == table[index].num) {
			map[k].count++;
			exists = 1;
			break;
		}

	if (exists == 0) {
		map[maplen].num = pages[i];
		map[maplen].count = 1;
		maplen++;
	}

	table[index].index = count;
	count++;
	faults++;

	for (int j=0; j<m; j++) {
		if (free) {
			if (j <= index)
				printf("%d ", table[j].num);
			else
				printf(" ");
```

```
                        } else
                                printf("%d ", table[j].num);
                    }
                    printf("\n");
                }
        }

        printf("\nNo of page faults = %d\n", faults);
}

void main() {
        int m, n, opt;

        printf("Enter the page table capacity: ");
        scanf("%d", &m);

        printf("Enter the no of page requests: ");
        scanf("%d", &n);

        int pages[n];

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

        while(1) {
                printf("\n1. FIFO\n2. LRU\n3. LFU\n4. Exit");
                printf("\nChoose option: ");
                scanf("%d", &opt);

                switch (opt) {
                        case 1:
                                fifo(m, n, pages);
                                break;
                        case 2:
                                lru(m, n, pages);
                                break;
                        case 3:
                                lfu(m, n, pages);
                                break;
                        case 4:
                                printf("\nExit.\n");
                                exit(0);
                        default:
                                printf("\nInvalid option!\n");
                }
        }
}
```

SAMPLE OUTPUT

```
Enter the page table capacity: 3
Enter the no of page requests: 16
Enter the page requests:
7
0
```

1
2
0
3
0
4
2
3
0
3
2
1
2
0

1. FIFO
2. LRU
3. LFU
4. Exit
Choose option: 1

FIFO
7: 7
0: 7
1: 7 1
2: 7 1 2
0: 0 1 2
3: 0 3 2
0: 0 3 2
4: 0 3 4
2: 2 3 4
3: 2 3 4
0: 2 0 4
3: 2 0 3
2: 2 0 3
1: 1 0 3
2: 1 2 3
0: 1 2 0

No of page faults = 12

1. FIFO
2. LRU
3. LFU
4. Exit
Choose option: 2

LRU
7: 7
0: 7
1: 7 1
2: 7 1 2
0: 0 1 2
3: 0 3 2
0: 0 3 2
4: 0 3 4
2: 0 2 4

3: 3 2 4
0: 3 2 0
3: 3 2 0
2: 3 2 0
1: 3 2 1
2: 3 2 1
0: 0 2 1

No of page faults = 11

1. FIFO
2. LRU
3. LFU
4. Exit
Choose option: 3

LFU
7: 7
0: 7
1: 7 1
2: 7 1 2
0: 0 1 2
3: 0 3 2
0: 0 3 2
4: 0 3 4
2: 0 2 4
3: 0 2 3
0: 0 2 3
3: 0 2 3
2: 0 2 3
1: 1 2 3
2: 1 2 3
0: 0 2 3

No of page faults = 10

1. FIFO
2. LRU
3. LFU
4. Exit
Choose option: 4

Exit.