Aim:

To Implement Page Replacement policies like LRU, Optimal Replacement Policy, MFU,FIFO

Algorithm:

FIFO:

1.Iterate through the sequence and check if char found in page table

2.If not increment miss variable set the Fifo Pointer to the character

3.Print the total page faults

LFU:

1.Maintain a miss, in and index variable

2.Maintain a frequency table that tracks previous requests

3.Replace the one that was least used

4.Print the total page Faults

Optimal Replacement Policy:

1.Maintain a miss,index variable

2.Check for Future Occurrence for all the elements in the page table

3.Replace one that has the least of all future occurrences as the need arises.

4.Print the total page faults

MFU:

1. Maintain a miss,index variable

2. Maintain a frequency table that tracks previous requests

3. Replace the one that was most used as the need arises

4. Print the total page Faults

Code:

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

int isFound(char * arr,int length,char seq){
    for(int i=0;i<length;i++){
```

```
		if(seq==*(arr+i)){
			return 1;
		}
	}
	return 0;

}
void FIFO(char* table,char* arr,int n_pages,int length){
	//printf("%s",arr);
	int miss=0,ptr=0;
	for(int i=0;i<=length;i++){
		if(isFound(table,n_pages,arr[i])==0){
			miss++;//printf("55");
			*(table+((ptr++)%n_pages))=*(arr+i);
		}
		for(int k=0;k<n_pages;k++){
			printf("%c\t",table[k]);
		}
		printf("\n=======\n");
		}

	printf("\nMiss %d\n",miss);
}
int FindNextOccurrence(char* arr,int length,char charecter,int source){
	int val=0;
	//if(!isFound(table,n_pages,charecter)){
	//}
	for(int i=source;i<length;i++){
		val++;
		if(charecter==*(arr+i)){
			break;
```

```
                }
        }
        return val;
}
int min(int *arr,int n_pages){
        int max=*(arr+0),maxidx=0;
        for(int i=0;i<n_pages;i++){
                if(*(arr+i)>max){
                        max=*(arr+i);
                        maxidx=i;

                }
        }
        return maxidx;
}

void  OPTIMAL_REPLACEMENT_POLICY(char* table,char* arr,int n_pages,int length){
        int miss=0;
        int tmp[n_pages];
        int  max=0 ,idx=0;
        for(int i=0;i<length;i++){
                max=0;

                for(int k=0;k<n_pages;k++){
                        printf("%c\t",table[k]);
                }
                printf("\n=======\n");
                if(isFound(table,n_pages,*(arr+i))==1){
                        continue;
                }
                for(int j=0;j<n_pages;j++){
```

```
                        if(table[j]=='n'){
                                //miss++;
                                *(table+j)=*(arr+i);
                                break;
                        }
                        //printf("jj%d\n",FindNextOccurrence(arr,length,table[j],i));
                        if(FindNextOccurrence(arr,length,table[j],i)>max){
                                max=FindNextOccurrence(arr,length,table[j],i);
                                idx=j;
                                //printf("\n%d\n",idx);
                        }
                }
                miss++;
                table[idx]=*(arr+i);

        }
        printf("\n %d \n",miss);
}
int FreqPrevious(char* arr, int len ,char charecter,int source){
        int val=0;
        for(int i=source ;i>=0;i--){

                if(charecter==*(arr+i)){
                        val++;
                }
        }
        return val;


}
void LFUPolicy(char* table,char*arr,int n_pages,int length){
```

```
	int miss=0;
	int min=100;
	int idx=0;
	for(int i=0;i<length;i++){
		min=100;
		for(int k=0;k<n_pages;k++){
			printf("%c\t",table[k]);
		}
		printf("\n=======\n");
		if(isFound(table,n_pages,*(arr+i))){
			continue;
		}
		for(int j=0;j<n_pages;j++){
			if(*(table+j)==-1){
				//miss++;
				*(table+j)=*(arr+i);
				break;
			}
			if(FreqPrevious(arr,length,table[j],i)<min){
				min=FreqPrevious(arr,length,table[j],i);
				idx=j;
			}
		}
		miss++;
		table[idx]=*(arr+i);
	}
	printf("\nMiss%d\n",miss);
}
void MFUPolicy(char* table,char*arr,int n_pages,int length){
	int miss=0;
	int max=0;
```

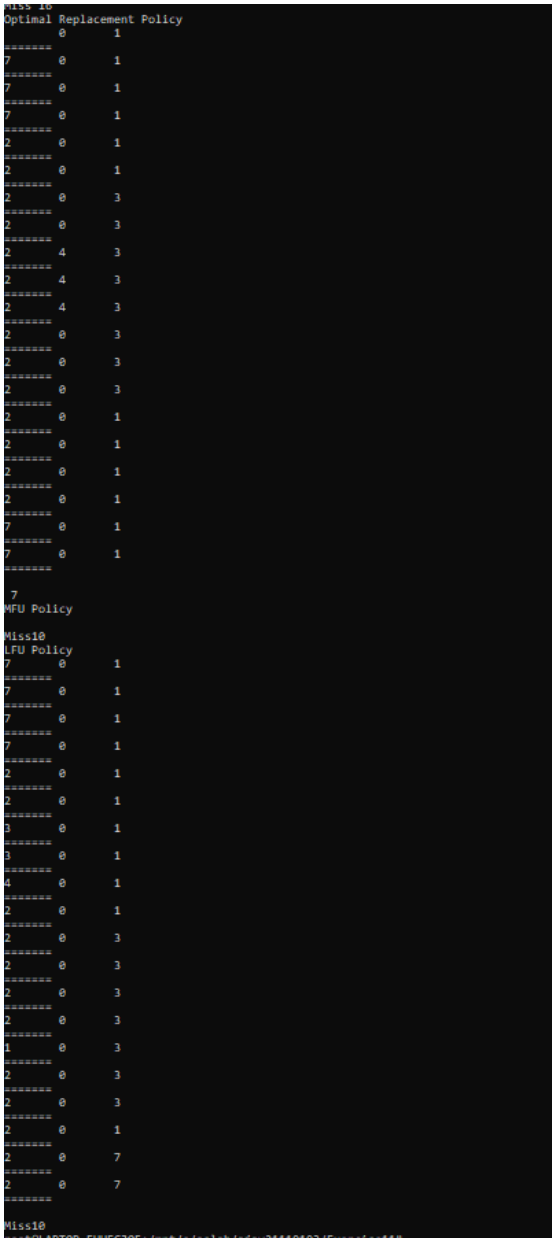
```
	int idx=0;
	for(int i=0;i<length;i++){
		max=0;
		if(isFound(table,n_pages,*(arr+i))){
			continue;
		}
		for(int j=0;j<n_pages;j++){
			if(*(table+j)==-1){
				//miss++;
				*(table+j)=*(arr+i);
				break;
			}
			if(FreqPrevious(arr,length,table[j],i)>max){
				max=FreqPrevious(arr,length,table[j],i);
				idx=j;
			}
		}
		miss++;
		*(table+idx)=*(arr+i);
	}
	printf("\nMiss%d\n",miss);
}
int main(){
	int n,n_pages;
	scanf("%d %d",&n,&n_pages);
	char *table=(char*)malloc(n_pages*sizeof(char));
	char *a=(char*)malloc(n*sizeof(char));
	scanf(" %s",a);
	for(int i=0;i<n_pages;i++){*(table+i)='n';}
	printf("%d",isFound(a,n,'3'));
	FIFO(table,a,n_pages,n);
```

```
        OPTIMAL_REPLACEMENT_POLICY(table,a,n_pages,n);
        MFUPolicy(table,a,n_pages,n);
        LFUPolicy(table,a,n_pages,n);
        printf("==%d",FindNextOccurrence(a,n,'7',3));
}
```

Output:

```
root@LAPTOP-FHHEGJQ5:/mnt/e/oslab/ajay21110103/Exercise11# nano PageTablenew.c
root@LAPTOP-FHHEGJQ5:/mnt/e/oslab/ajay21110103/Exercise11# gcc PageTablenew.c
root@LAPTOP-FHHEGJQ5:/mnt/e/oslab/ajay21110103/Exercise11# ./a.out
20 3
70120304230321201701
FIFO
7       n       n
=======
7       0       n
=======
7       0       1
=======
2       0       1
=======
2       0       1
=======
2       3       1
=======
2       3       0
=======
4       3       0
=======
4       2       0
=======
4       2       3
=======
0       2       3
=======
0       2       3
=======
0       2       3
=======
0       1       3
=======
0       1       2
=======
0       1       2
=======
0       1       2
=======
7       1       2
=======
7       0       2
=======
7       0       1
=======
        0       1
=======

Miss 16
```



Result:

Thus the Page Replacement Algorithms were implemented and their efficacies were studied.