EXPERIMENT 11
PAGE REPLACEMENT ALGORITHMS

AIM
Write C program to simulate page replacement Algorithms FIFO, LRU & LFU.

FIFO ALGORITHM
STEP 0: START
STEP 1: Declare all the variables required for this algorithm
i & j - Loop counters
n & m - number of pages & frames
found - Flag to indicate if page is found in the frame.
page[100] - Array to store page numbers
frame[100] - Array to represent frames
k - Index for frame replacement
count - counter for page faults.
STEP 2: Prompt the user to input the number of pages
STEP 3: Prompt the user to input the page numbers one by one using for loop
STEP 4: Prompt user to input number of frames
STEP 5: Initialize frame array with -1 indicating empty frames

STEP 6: Check if page is already in frame (frame[j] == page[i]), if found, then set found = 1.
STEP 7: If page is not found, replace a page in the frame by setting frame[k] = page[i]
STEP 8: Finally display the frames and specify HIT/MISS depending upon found = 1 or found = 0.
STEP 9: Print the amount of page faults by count variable.
STEP 10: END

ALGORITHM FOR LRU
STEP 0: START
STEP 1: Define a structure frames with two members: content to hold the page number and count to keep track of when the page was last referenced. An array of frames is declared using structure to represent the frames.
STEP 2: Variables are declared -
i, j & k - Loop counters
p & f - Number of pages & frames
page[100] - Array to store page numbers
cnt - counter for assigning reference number to pages
min - variable to store the index of

frame with minimum count.
pf - counter for page faults
id - Index for managing frames.
STEP 3: Prompt the user to input the number of pages and the referencing string.
STEP 4: Prompt user to input number of frames.
STEP 5: Initialize frame[i].content=-1 & frame[i].count=0
STEP 6: The outer loop iterates over each page in the reference string
STEP 7: The inner loop checks if page is already present in any of the frames. If found ('HIT') and then updates the count for the frame and breaks out of loop.
STEP 8: If page not found ('MISS') then, it replaces a frame using the logic-
i) If empty frame available then, page is placed in that frame.
ii) If all frames are occupied then, it selects the frame that will not be used for the longest time in the future based on reference string
STEP 9: After each iteration, it prints the status (HIT/MISS) and content of the frames.

STEP 10: Print the total number of page faults that occurred during the execution of the algorithm
STEP 11: END

ALGORITHM FOR LFU
STEP 0: START
STEP 1: Define a structure named frame with three members:
content - to hold page number
freq - to store frequency of page references
count - To keep track of number of page references.
An array of structures frames is declared to represent frames.
STEP 2: Variables are declared -
i & j - LOOP counters
pg - Number of pages
fr - Number of frames
count - counter for assigning number of page references.
pf - counter for page faults
min - Variable to store index of frame with minimum frequency & minimum count
page[100] - Array to store page numbers
id - Index for managing frames.

STEP 3: Prompt the user to input the number of pages & the referencing string.
STEP 4: Prompt the user to input the number of frames.
STEP 5: Initialize frames[i].content = -1 frames[i].freq = 0 & frames[i].count = 0
STEP 6: The outer loop iterates over each page in the reference string
STEP 7: The inner loop checks if the page is already present in any of the frames. If found ('HIT') and increment frequency of that frame and breaks out of loop.
STEP 8: If page not found ('MISS'), it replaces a based on the combined criteria of least frequency & least recently used:
i) If there are empty frames available, it placed in one of them
ii) If all frames are occupied, it selects frame with lowest frequency. In case of tie, it selects the frame that was least recently used (lowest count)
iii) After each iteration, it prints the status of the page (HIT/MISS) and content of the frames.
iv) It also updates the page fault

count ('Pf')
STEP 9: Print the total number of page faults that occurred during the execution of the algorithm.
STEP 10: END.

RESULT
Exphiment executed successfully and output obtained.