Assignment-06

Code :

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

typedef struct {
    char data[20][2];
    int end;
} queue;

void enqueue(queue *q, char data[], int position) {
    strncpy(q->data[position], data, 2);
}

char* dequeue(queue *q, int position) {
    return q->data[position];
}

void fifo(char string[], int frameSize, int count, int* pageHits) {
    int cnt, cnt2, flag, faults = 0;
    queue q;
    int firstin = -1;
    q.end = 0;

    printf("\nData Requested\tFrame contents\t    Page
Fault\n=============================================");

    for (cnt = 0; cnt < count; cnt += 2) {
        printf("\n\n\t%c", string[cnt]);
        flag = 0;

        for (cnt2 = 0; cnt2 < q.end; cnt2++) {
            if (string[cnt] == q.data[cnt2][0]) {
                flag = 1;
                break;
            }
        }

        if (flag == 0) {
            faults++;

            if (q.end < frameSize) {
                enqueue(&q, string + cnt, q.end);
```

```
            q.end++;
        } else {
            dequeue(&q, firstin);
            firstin = (firstin + 1) % q.end;
            enqueue(&q, string + cnt, firstin);
        }

        printf("\t  ");

        for (cnt2 = 0; cnt2 < q.end; cnt2++) {
            printf("%c   ", q.data[cnt2][0]);
        }

        printf("\t\tY");
    } else {
        pageHits[0]++;
        printf("\t  ");

        for (cnt2 = 0; cnt2 < q.end; cnt2++) {
            printf("%c   ", q.data[cnt2][0]);
        }

        printf("\t\tN");
    }
  }

  printf("\n\n=============================================\n");
  printf("\nTotal no. of Page Faults: %d\n", faults);
  printf("Page Hit Ratio: %.2f\n", (float)pageHits[0] / count);
}

void optimal(char string[], int frameSize, int count, int* pageHits) {
  int cnt, cnt2, selector, flag, max, faults = 0;
  int distance[20];
  queue q;
  q.end = 0;

  printf("\nData Requested\tFrame contents\t    Page
Fault\n=============================================");

  for (cnt = 0; cnt < count; cnt += 2) {
    printf("\n\n\t%c", string[cnt]);
    flag = 0;

    for (cnt2 = 0; cnt2 < q.end; cnt2++) {
      if (string[cnt] == q.data[cnt2][0]) {
        flag = 1;
        break;
```

```
    }
}

if (flag == 0) {
    faults++;

    if (q.end < frameSize) {
        enqueue(&q, string + cnt, q.end);
        q.data[q.end][1] = cnt;
        q.end++;
    } else {
        for (cnt2 = 0; cnt2 < q.end; cnt2++) {
            distance[cnt2] = 0;
        }

        for (selector = 0; selector < q.end; selector++) {
            for (cnt2 = cnt; cnt2 < count; cnt2 += 2) {
                if (string[cnt2] == q.data[selector][0]) {
                    distance[selector] = cnt2 / 2;
                    break;
                }

                if (distance[selector] == 0) {
                    distance[selector] = 99 - q.data[selector][1];
                }
            }
        }

        max = 0;

        for (cnt2 = 0; cnt2 < q.end; cnt2++) {
            if (distance[cnt2] > max) {
                max = distance[cnt2];
                selector = cnt2;
            }
        }

        dequeue(&q, selector);
        enqueue(&q, string + cnt, selector);
        q.data[selector][1] = cnt;
    }

    printf("\t  ");

    for (cnt2 = 0; cnt2 < q.end; cnt2++) {
        printf("%c   ", q.data[cnt2][0]);
    }
```

```
            printf("\t\tY");
        } else {
            pageHits[1]++;
            printf("\t  ");

            for (cnt2 = 0; cnt2 < q.end; cnt2++) {
                printf("%c   ", q.data[cnt2][0]);
            }

            printf("\t\tN");
        }
    }

    printf("\n\n================================================\n");
    printf("\nTotal no. of Page Faults: %d\n", faults);
    printf("Page Hit Ratio: %.2f\n", (float)pageHits[1] / count);
}

void lru(char string[], int frameSize, int count, int* pageHits) {
    int cnt, cnt2, selector, flag, min, faults = 0;
    queue q;
    q.end = 0;

    printf("\nData Requested\tFrame contents\t    Page
Fault\n================================================");

    for (cnt = 0; cnt < count; cnt += 2) {
        printf("\n\n\t%c", string[cnt]);
        flag = 0;

        for (cnt2 = 0; cnt2 < q.end; cnt2++) {
            if (string[cnt] == q.data[cnt2][0]) {
                q.data[cnt2][1] = (cnt / 2) + 1;
                flag = 1;
                break;
            }
        }

        if (flag == 0) {
            faults++;

            if (q.end < frameSize) {
                enqueue(&q, string + cnt, q.end);
                q.data[q.end][1] = (cnt / 2) + 1;
                q.end++;
            } else {
                min = 99;
```

```
            for (cnt2 = 0; cnt2 < q.end; cnt2++) {
                if (q.data[cnt2][1] < min) {
                    min = q.data[cnt2][1];
                    selector = cnt2;
                }
            }

            dequeue(&q, selector);
            enqueue(&q, string + cnt, selector);
            q.data[selector][1] = (cnt / 2) + 1;
        }

        printf("\t  ");

        for (cnt2 = 0; cnt2 < q.end; cnt2++) {
            printf("%c   ", q.data[cnt2][0]);
        }

        printf("\t\tY");
    } else {
        pageHits[2]++;
        printf("\t  ");

        for (cnt2 = 0; cnt2 < q.end; cnt2++) {
            printf("%c   ", q.data[cnt2][0]);
        }

        printf("\t\tN");
    }
}

printf("\n\n=============================================\n");
printf("\nTotal no. of Page Faults: %d\n", faults);
printf("Page Hit Ratio: %.2f\n", (float)pageHits[2] / count);
}

int main() {
    int frameSize, count, ch;
    char string[51];
    int pageHits[3] = {0, 0, 0}; // 0: FIFO, 1: Optimal, 2: LRU

    printf("Enter the string: ");
    count = 0;

    do {
        scanf("%c", &string[count]);
        count++;
    } while (string[count - 1] != '\n');
```

```
    count--;

    printf("\nEnter the size of the frame: ");
    scanf("%d", &frameSize);

    do {
        printf("\nMENU\n====\n1.FIFO\n2.Least Recently Used
(LRU)\n3.Optimal\n4.Exit\n\nYour Choice:");
        scanf("%d", &ch);

        switch (ch) {
            case 1:
                fifo(string, frameSize, count, pageHits);
                break;

            case 2:
                lru(string, frameSize, count, pageHits);
                break;

            case 3:
                optimal(string, frameSize, count, pageHits);
                break;

            case 4:
                // exit(0);
                break;

            default:
                printf("\nInvalid choice! Please try again!");
                continue;
        }
    } while (ch != 4);

    printf("\nOverall Page Hit Ratio for FIFO: %.2f\n", (float)pageHits[0] / count);
    printf("Overall Page Hit Ratio for Optimal: %.2f\n", (float)pageHits[1] / count);
    printf("Overall Page Hit Ratio for LRU: %.2f\n", (float)pageHits[2] / count);

    return 0;
}
```

Output :

Enter the string: 4761761272

Enter the size of the frame: 3

MENU
====
1.FIFO
2.Least Recently Used (LRU)
3.Optimal
4.Exit

Your Choice:1

Data Requested Frame contents Page Fault
==========

4 4 Y

6 4 6 Y

7 4 6 7 Y

1 1 6 7 Y

7 1 6 7 N

==========

Total no. of Page Faults: 4
Page Hit Ratio: 0.10

MENU
====
1.FIFO
2.Least Recently Used (LRU)
3.Optimal
4.Exit

Your Choice:2

Data Requested Frame contents Page Fault
==========

4 4 Y

6 4 6 Y

7 4 6 7 Y

1 1 6 7 Y

```
    7      1  6  7         N

================================================

Total no. of Page Faults: 4
Page Hit Ratio: 0.10

MENU
====
1.FIFO
2.Least Recently Used (LRU)
3.Optimal
4.Exit

Your Choice:3

Data Requested  Frame contents     Page Fault
================================================

    4      4         Y

    6      4  6           Y

    7      4  6  7         Y

    1      1  6  7         Y

    7      1  6  7         N

================================================

Total no. of Page Faults: 4
Page Hit Ratio: 0.10

MENU
====
1.FIFO
2.Least Recently Used (LRU)
3.Optimal
4.Exit

Your Choice:4

Overall Page Hit Ratio for FIFO: 0.10
Overall Page Hit Ratio for Optimal: 0.10
```

Overall Page Hit Ratio for LRU: 0.10