YUVANESH KS

OS LAB BATCH-2 ANSWER

Q7: Deadlock Avoidance using Banker's Algorithm (C Program)

```
#include <stdio.h>

int main() {
    int alloc[5][3] = {{0,1,0},{2,0,0},{3,0,2},{2,1,1},{0,0,2}};
    int max[5][3] = {{7,5,3},{3,2,2},{9,0,2},{2,2,2},{4,3,3}};
    int avail[3] = {3,3,2};
    int need[5][3], finish[5] = {0}, safeSeq[5];
    int i, j, k, count = 0;

    for(i = 0; i < 5; i++)
        for(j = 0; j < 3; j++)
            need[i][j] = max[i][j] - alloc[i][j];

    while(count < 5) {
        for(i = 0; i < 5; i++) {
            if(!finish[i]) {
                for(j = 0; j < 3; j++)
                    if(need[i][j] > avail[j]) break;
                if(j == 3) {
                    for(k = 0; k < 3; k++) avail[k] += alloc[i][k];
                    safeSeq[count++] = i;
                    finish[i] = 1;
                }
            }
        }
    }

    printf("Safe sequence is: ");
    for(i = 0; i < 5; i++) printf("P%d ", safeSeq[i]);
    return 0;
}
```

Q8: a) Shell Script - Check Palindrome

```
#!/bin/bash
echo "Enter a string:"
read str
rev=$(echo $str | rev)
if [ "$str" = "$rev" ]; then
  echo "Palindrome"
else
  echo "Not Palindrome"
fi
```

Q8: b) C Program - First Fit Memory Allocation

```
#include <stdio.h>

int main() {
    int blockSize[] = {100, 500, 200, 300, 600};
```

```c
    int processSize[] = {212, 417, 112, 426};
    int allocation[4], i, j;

    for(i = 0; i < 4; i++) allocation[i] = -1;

    for(i = 0; i < 4; i++) {
        for(j = 0; j < 5; j++) {
            if(blockSize[j] >= processSize[i]) {
                allocation[i] = j;
                blockSize[j] -= processSize[i];
                break;
            }
        }
    }

    for(i = 0; i < 4; i++) {
        printf("Process %d -> Block %d\n", i+1, (allocation[i] != -1) ?
allocation[i]+1 : -1);
    }

    return 0;
}
```

Q9: a) Shell Script - Case Statement for UNIX Commands

```bash
#!/bin/bash
echo "1. Date"
echo "2. List files"
echo "3. Present directory"
read -p "Choose an option: " choice

case $choice in
  1) date ;;
  2) ls ;;
  3) pwd ;;
  *) echo "Invalid option" ;;
esac
```

Q9: b) C Program - SSTF Disk Scheduling

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

int main() {
    int n = 8, head = 50;
    int request[] = {82, 170, 43, 140, 24, 16, 190};
    int i, j, min, pos, total = 0, visited[8] = {0};

    for(i = 0; i < n; i++) {
        min = 9999;
        for(j = 0; j < n; j++) {
            if(!visited[j] && abs(head - request[j]) < min) {
                min = abs(head - request[j]);
                pos = j;
            }
        }
        total += abs(head - request[pos]);
        head = request[pos];
        visited[pos] = 1;
    }

    printf("Total seek time: %d\n", total);
    return 0;
}
```

Q12: a) C Program - Page Replacement (FIFO)

```
#include <stdio.h>

int main() {
    int pages[] = {1, 3, 0, 3, 5, 6};
    int n = 6, frames = 3, i, j, k = 0, faults = 0, queue[3] = {-1, -1, -1};

    for(i = 0; i < n; i++) {
        int found = 0;
        for(j = 0; j < frames; j++) {
            if(queue[j] == pages[i]) {
                found = 1;
                break;
            }
        }
        if(!found) {
            queue[k] = pages[i];
            k = (k + 1) % frames;
            faults++;
        }
    }

    printf("Page faults: %d\n", faults);
    return 0;
}
```

Q14: b) Shell Script - Best Fit Memory Allocation

```
#!/bin/bash
blocks=(100 500 200 300 600)
processes=(212 417 112 426)

for i in ${!processes[@]}; do
    best=-1
    for j in ${!blocks[@]}; do
        if [ ${blocks[j]} -ge ${processes[i]} ]; then
            if [ $best -eq -1 ] || [ ${blocks[j]} -lt ${blocks[best]} ]; then
                best=$j
            fi
        fi
    done
    if [ $best -ne -1 ]; then
        echo "Process ${processes[i]}KB -> Block ${blocks[best]}KB"
        blocks[best]=$((blocks[best]-processes[i]))
    else
        echo "Process ${processes[i]}KB -> Not Allocated"
    fi
done
```

Q15: b) C Program - Indexed File Allocation

```
#include <stdio.h>

int main() {
    int indexBlock = 5, blocks[] = {10, 11, 12, 13}, n = 4;

    printf("Index Block: %d\n", indexBlock);
    printf("Blocks: ");
    for(int i = 0; i < n; i++) {
        printf("%d ", blocks[i]);
    }
    printf("\nFile stored using Indexed Allocation.\n");

    return 0;
}
```

Q16: a) Shell Script - Sum and Average of 4 Integers

```
#!/bin/bash
sum=0
echo "Enter 4 numbers:"
for i in {1..4}
do
    read num
    sum=$((sum + num))
done
avg=$((sum / 4))
echo "Sum = $sum"
echo "Average = $avg"
```

Q17: C Program - SCAN Disk Scheduling

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

int main() {
    int i, j, n = 8, head = 50, max = 200, direction = 1;
    int request[] = {82, 170, 43, 140, 24, 16, 190};
    int seek = 0, temp, disk[100];

    for(i = 0; i < n; i++) disk[i] = request[i];
    disk[n++] = head;
    if(direction) disk[n++] = max - 1;
    else disk[n++] = 0;
```

```c
    for(i = 0; i < n-1; i++)
        for(j = i+1; j < n; j++)
            if(disk[i] > disk[j]) {
                temp = disk[i]; disk[i] = disk[j]; disk[j] = temp;
            }

    for(i = 0; i < n; i++) if(disk[i] == head) break;
    if(direction)
        for(j = i; j < n-1; j++) seek += abs(disk[j+1] - disk[j]);
    else
        for(j = i; j > 0; j--) seek += abs(disk[j] - disk[j-1]);

    printf("Total seek time: %d\n", seek);
    return 0;
}
```

Q18: a) UNIX Command to Print Length of Longest Line

```
awk '{ if (length > max) max = length } END { print max }' filename.txt
```

Q19: b) C Program - LRU Page Replacement

```c
#include <stdio.h>

int findLRU(int time[], int n) {
    int i, min = time[0], pos = 0;
    for(i = 1; i < n; i++) {
        if(time[i] < min) {
            min = time[i];
            pos = i;
        }
    }
    return pos;
}

int main() {
    int pages[] = {7, 0, 1, 2, 0, 3, 0, 4, 2, 3};
    int frames[3], time[3], counter = 0, faults = 0, i, j, pos, flag;

    for(i = 0; i < 10; i++) {
        flag = 0;
        for(j = 0; j < 3; j++) {
            if(frames[j] == pages[i]) {
                counter++;
                time[j] = counter;
                flag = 1;
                break;
            }
        }
        if(!flag) {
            pos = findLRU(time, 3);
            frames[pos] = pages[i];
            counter++;
            time[pos] = counter;
            faults++;
        }
    }

    printf("Page Faults: %d\n", faults);
    return 0;
}
```

Q20: C Program – LOOK Disk Scheduling

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

void sort(int arr[], int n) {
    int i, j, temp;
    for(i = 0; i < n - 1; i++)
        for(j = i + 1; j < n; j++)
            if(arr[i] > arr[j]) {
                temp = arr[i];
                arr[i] = arr[j];
                arr[j] = temp;
            }
}

int main() {
    int requests[] = {82, 170, 43, 140, 24, 16, 190};
    int n = 7, head = 50, i, j, seek = 0;
    int up[10], down[10], upCount = 0, downCount = 0;

    for(i = 0; i < n; i++) {
        if(requests[i] >= head)
            up[upCount++] = requests[i];
        else
            down[downCount++] = requests[i];
    }

    sort(up, upCount);
    sort(down, downCount);

    // Move up
    for(i = 0; i < upCount; i++) {
        seek += abs(head - up[i]);
        head = up[i];
    }

    // Then move down
    for(i = downCount - 1; i >= 0; i--) {
        seek += abs(head - down[i]);
        head = down[i];
    }
    printf("Total Seek Time (LOOK): %d\n", seek);
    return 0;
}
```

Q23: C Program – C-SCAN Disk Scheduling

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

void sort(int arr[], int n) {
    for(int i = 0; i < n - 1; i++)
        for(int j = i + 1; j < n; j++)
            if(arr[i] > arr[j]) {
                int temp = arr[i];
                arr[i] = arr[j];
                arr[j] = temp;
            }
}

int main() {
    int request[] = {82, 170, 43, 140, 24, 16, 190};
    int n = 7, head = 50, disk_size = 200;
    int seek = 0, i, index;

    sort(request, n);

    for(i = 0; i < n; i++)
        if(request[i] > head) break;

    // Move toward higher end
    for(int j = i; j < n; j++) {
        seek += abs(head - request[j]);
        head = request[j];
    }

    // Jump to start
    if(i > 0) {
        seek += abs(head - (disk_size - 1));
        head = 0;
        seek += abs(head - request[0]);

        for(int j = 1; j < i; j++) {
            seek += abs(head - request[j]);
            head = request[j];
        }
    }

    printf("Total Seek Time (C-SCAN): %d\n", seek);
    return 0;
}
```

Q24: C Program – C-LOOK Disk Scheduling

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

void sort(int arr[], int n) {
    for(int i = 0; i < n - 1; i++)
        for(int j = i + 1; j < n; j++)
            if(arr[i] > arr[j]) {
                int temp = arr[i];
                arr[i] = arr[j];
                arr[j] = temp;
            }
}

int main() {
    int request[] = {82, 170, 43, 140, 24, 16, 190};
    int n = 7, head = 50, seek = 0, i;

    sort(request, n);

    for(i = 0; i < n; i++)
        if(request[i] > head) break;

    // Move up
    for(int j = i; j < n; j++) {
        seek += abs(head - request[j]);
        head = request[j];
    }

    // Jump to the lowest
    if(i > 0) {
        seek += abs(head - request[0]);
        head = request[0];

        for(int j = 1; j < i; j++) {
            seek += abs(head - request[j]);
            head = request[j];
        }
    }

    printf("Total Seek Time (C-LOOK): %d\n", seek);
    return 0;
}
```