



PUNJAB ENGINEERING COLLEGE (DEEMED TO BE UNIVERSITY) CHANDIGARH



Assignment 7

Submitted By :

Sugam Arora

SID : 21105021

Branch : ECE

Date : 30th March, 2025

Implement Banker's Algorithm for Deadlock and implement algorithm for deadlock detection.

Implement Banker's Algorithm for Deadlock and implement algorithm for deadlock detection.

Banker.cpp

```
#include <iostream>
using namespace std;

const int P = 5; // Number of processes
const int R = 3; // Number of resources

int available[R] = {3, 3, 2};

int max[P][R] = {
    {7, 5, 3},
    {3, 2, 2},
    {9, 0, 2},
    {2, 2, 2},
    {4, 3, 3}
};

int allocation[P][R] = {
    {0, 1, 0},
    {2, 0, 0},
    {3, 0, 2},
    {2, 1, 1},
    {0, 0, 2}
};

int need[P][R];

void calculateNeed() {
    for (int i = 0; i < P; i++)
        for (int j = 0; j < R; j++)
            need[i][j] = max[i][j] - allocation[i][j];
}

bool isSafe() {
    bool finish[P] = {false};
    int work[R];

    for (int i = 0; i < R; i++)
        work[i] = available[i];
```

```

int safeSeq[P], count = 0;

while (count < P) {
    bool found = false;
    for (int p = 0; p < P; p++) {
        if (!finish[p]) {
            bool canAllocate = true;
            for (int r = 0; r < R; r++) {
                if (need[p][r] > work[r]) {
                    canAllocate = false;
                    break;
                }
            }

            if (canAllocate) {
                for (int r = 0; r < R; r++)
                    work[r] += allocation[p][r];
                safeSeq[count++] = p;
                finish[p] = true;
                found = true;
            }
        }
    }

    if (!found) {
        cout << "System is not in a safe state.\n";
        return false;
    }
}

cout << "System is in a safe state.\nSafe Sequence: ";
for (int i = 0; i < P; i++)
    cout << "P" << safeSeq[i] << " ";
cout << endl;
return true;
}

int main() {
    calculateNeed();
    isSafe();
    return 0;
}

```

Deadlock_detection.cpp

```
#include <iostream>
using namespace std;

const int P = 4; // processes
const int R = 3; // resources

int allocation[P][R] = {
    {0, 1, 0},
    {2, 0, 0},
    {3, 0, 3},
    {2, 1, 1}
};

int request[P][R] = {
    {0, 0, 0},
    {2, 0, 2},
    {0, 0, 0},
    {1, 0, 0}
};

int available[R] = {1, 0, 1};

bool detectDeadlock() {
    bool finish[P] = {false};
    int work[R];

    for (int i = 0; i < R; i++)
        work[i] = available[i];

    while (true) {
        bool progress = false;

        for (int i = 0; i < P; i++) {
            if (!finish[i]) {
                bool canFinish = true;
                for (int j = 0; j < R; j++) {
                    if (request[i][j] > work[j]) {
                        canFinish = false;
                        break;
                    }
                }
            }
        }
    }
}
```

```

        if (canFinish) {
            for (int j = 0; j < R; j++)
                work[j] += allocation[i][j];
            finish[i] = true;
            progress = true;
        }
    }
}

if (!progress) break;
}

bool deadlock = false;
for (int i = 0; i < P; i++) {
    if (!finish[i]) {
        cout << "⚠ Process P" << i << " is in deadlock.\n";
        deadlock = true;
    }
}

if (!deadlock)
    cout << "✅ No deadlock detected.\n";

return deadlock;
}

int main() {
    detectDeadlock();
    return 0;
}

```

Os_deadlock_manager.sh

```
#!/bin/bash
```

```
# Colors
```

```
GREEN='\033[0;32m'
```

```
RED='\033[0;31m'
```

```
CYAN='\033[1;36m'
```

```
YELLOW='\033[1;33m'
```

```
NC='\033[0m' # No Color
```

```
# Banner
```

```
echo -e "${CYAN}"
```

```
echo "=====
```

```
echo "   🧠 OS Deadlock Management Simulator"
```

```
echo "=====
```

```
echo -e "${NC}"
```

```
# Compile functions
```

```
compile_and_run_banker() {
```

```
    echo -e "${YELLOW}Compiling Banker's Algorithm...${NC}"
```

```
    g++ banker.cpp -o banker
```

```
    if [ $? -eq 0 ]; then
```

```
        echo -e "${GREEN}✓ Compilation successful.${NC}"
```

```
        echo -e "${CYAN}► Executing Banker's Algorithm:${NC}"
```

```
        ./banker
```

```
    else
```

```
        echo -e "${RED}✗ Compilation failed!${NC}"
```

```
    fi
```

```
}
```

```
compile_and_run_deadlock() {
```

```
    echo -e "${YELLOW}Compiling Deadlock Detection Algorithm...${NC}"
```

```
    g++ deadlock_detection.cpp -o deadlock
```

```
    if [ $? -eq 0 ]; then
```

```
        echo -e "${GREEN}✓ Compilation successful.${NC}"
```

```
        echo -e "${CYAN}► Executing Deadlock Detection Algorithm:${NC}"
```

```
        ./deadlock
```

```
    else
```

```
        echo -e "${RED}✗ Compilation failed!${NC}"
```

```
    fi
```

```
}
```

```
# Main menu
```

```

while true; do
    echo -e "${CYAN}"
    echo "Choose an option:"
    echo "1) Run Banker's Algorithm"
    echo "2) Run Deadlock Detection Algorithm"
    echo "3) Run Both"
    echo "4) Exit"
    echo -e "${NC}"
    read -p "Enter your choice [1-4]: " choice

    case $choice in
        1)
            compile_and_run_banker
            ;;
        2)
            compile_and_run_deadlock
            ;;
        3)
            compile_and_run_banker
            echo
            compile_and_run_deadlock
            ;;
        4)
            echo -e "${GREEN}Exiting... Have a deadlock-free day! 🚀 ${NC}"
            break
            ;;
        *)
            echo -e "${RED}Invalid choice. Please select 1-4. ${NC}"
            ;;
    esac
    echo -e "\n-----\n"
done

```