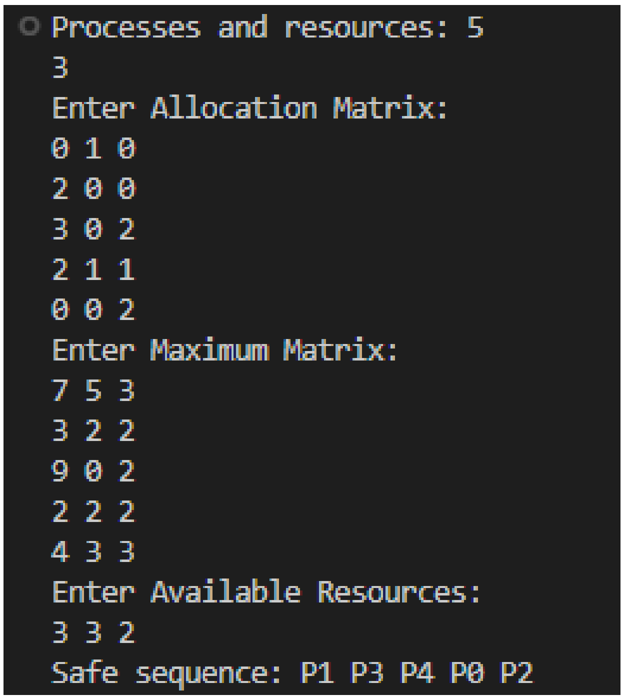
OS LAB5 – 8/05/25

1. Write a c program to stimulate bankers algorithem for the purpose of deadlock avoidance

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

int main() {
    int P, R;
    printf("Processes and resources: ");
    scanf("%d %d", &P, &R);

    int alloc[P][R], max[P][R], need[P][R], avail[R], safeSeq[P];
    bool finish[P];

    printf("Enter Allocation Matrix:\n");
    for (int i = 0; i < P; i++)
        for (int j = 0; j < R; j++)
            scanf("%d", &alloc[i][j]);

    printf("Enter Maximum Matrix:\n");
    for (int i = 0; i < P; i++)
        for (int j = 0; j < R; j++)
            scanf("%d", &max[i][j]);

    printf("Enter Available Resources:\n");
    for (int i = 0; i < R; i++)
        scanf("%d", &avail[i]);

    for (int i = 0; i < P; i++) {
        finish[i] = false;
        for (int j = 0; j < R; j++)
            need[i][j] = max[i][j] - alloc[i][j];
    }

    int count = 0;
    while (count < P) {
        bool found = false;
        for (int i = 0; i < P; i++) {
            if (!finish[i]) {
                bool canRun = true;
                for (int j = 0; j < R; j++)
                    if (need[i][j] > avail[j]) canRun = false;
                if (canRun) {
                    for (int j = 0; j < R; j++)
                    avail[j] += alloc[i][j];
                safeSeq[count++] = i;
                finish[i] = true;
                found = true;
            }
        }
    }
    if (!found) break;
}

if (count == P) {
    printf("Safe sequence: ");
    for (int i = 0; i < P; i++)
        printf("P%d ", safeSeq[i]);
    printf("\n");
} else {
    printf("System is NOT in a safe state.\n");
}

return 0;
}
```

OUTPUT:



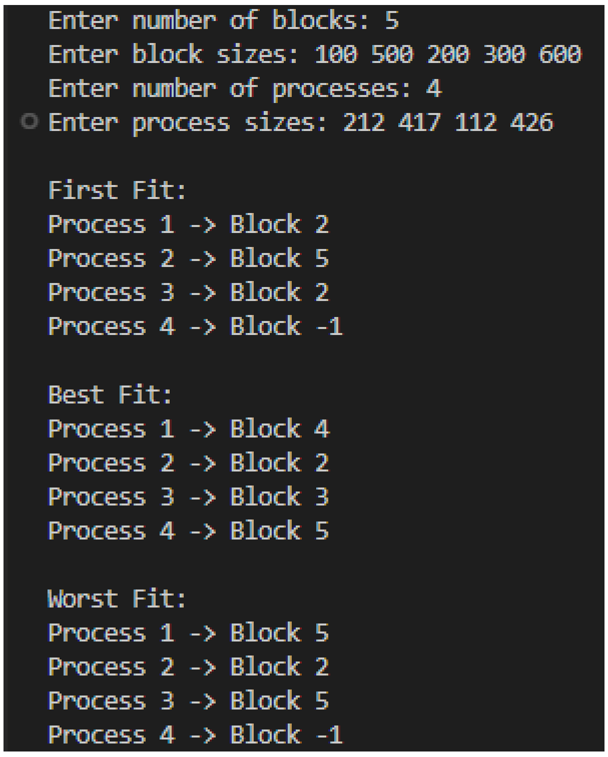
```
Processes and resources: 5
3
Enter Allocation Matrix:
0 1 0
2 0 0
3 0 2
2 1 1
0 0 2
Enter Maximum Matrix:
7 5 3
3 2 2
9 0 2
2 2 2
4 3 3
Enter Available Resources:
3 3 2
Safe sequence: P1 P3 P4 P0 P2
```

2. Write a c program to stimulate the following contiguous memory allocation techniques.
   A) worst-fit
   B) best-fit
   C) First-fit

```
#include <stdio.h>
#define MAX 100

void allocate(int blocks[], int n, int procs[], int m, char *type) {
   int alloc[m], i, j, idx;
   for (i = 0; i < m; i++) alloc[i] = -1;

   for (i = 0; i < m; i++) {
      idx = -1;
      for (j = 0; j < n; j++) {
         if (blocks[j] >= procs[i]) {
            if ((type[0] == 'F' && idx == -1) ||
               (type[0] == 'B' && (idx == -1 || blocks[j] < blocks[idx])) ||
               (type[0] == 'W' && (idx == -1 || blocks[j] > blocks[idx])))
               idx = j;
         }
      }
      if (idx != -1) {
         alloc[i] = idx;
         blocks[idx] -= procs[i];
      }
   }

   printf("\n%s Fit:\n", type);
   for (i = 0; i < m; i++)
      printf("Process %d -> Block %d\n", i + 1, alloc[i] == -1 ? -1 : alloc[i] + 1);
}

int main() {
   int n, m, i, blocks[MAX], procs[MAX], b1[MAX], b2[MAX];

   printf("Enter number of blocks: "); scanf("%d", &n);
   printf("Enter block sizes: "); for (i = 0; i < n; i++) scanf("%d", &blocks[i]);

   printf("Enter number of processes: "); scanf("%d", &m);
   printf("Enter process sizes: "); for (i = 0; i < m; i++) scanf("%d", &procs[i]);

   for (i = 0; i < n; i++) b1[i] = b2[i] = blocks[i];
```

```
    allocate(blocks, n, procs, m, "First");
    allocate(b1, n, procs, m, "Best");
    allocate(b2, n, procs, m, "Worst");

    return 0;
}
```

OUTPUT:



```
Enter number of blocks: 5
Enter block sizes: 100 500 200 300 600
Enter number of processes: 4
Enter process sizes: 212 417 112 426

First Fit:
Process 1 -> Block 2
Process 2 -> Block 5
Process 3 -> Block 2
Process 4 -> Block -1

Best Fit:
Process 1 -> Block 4
Process 2 -> Block 2
Process 3 -> Block 3
Process 4 -> Block 5

Worst Fit:
Process 1 -> Block 5
Process 2 -> Block 2
Process 3 -> Block 5
Process 4 -> Block -1
```