Name:Karan Gadekar
Roll:A-57
Program:

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

int main() {
    int pid;
    pid = fork();

    if (pid < 0) {
        printf("\n Error ");
        exit(1);
    } else if (pid == 0) {
        printf("\n Hello, I am the child process ");
        printf("\n My pid is %d ", getpid());
        exit(0);
    } else {
        printf("\n Hello, I am the parent process ");
        printf("\n My actual pid is %d \n ", getpid());
        exit(1);
    }
}
```

Output:

```
karan@karan-VirtualBox:~$ gcc example.c -o example
karan@karan-VirtualBox:~$ ./example

 Hello, I am the parent process
 My actual pid is 34605

 Hello, I am the child process
 My pid is 34606
karan@karan-VirtualBox:~$ ps
    PID TTY          TIME CMD
  28669 pts/3    00:00:00 bash
  34630 pts/3    00:00:00 ps
karan@karan-VirtualBox:~$
```

Name:Karan Gadekar
Roll:A-57
Program:

```
#include <stdio.h>

int main() {
    int n, m, i, j, k;
    n = 5; // Number of processes
    m = 3; // Number of resources

    // Allocation Matrix
    int alloc[5][3] = {
        {0, 1, 0}, // P0
        {2, 0, 0}, // P1
        {3, 0, 2}, // P2
        {2, 1, 1}, // P3
        {0, 0, 2}  // P4
    };

    // MAX Matrix
    int max[5][3] = {
        {7, 5, 3}, // P0
        {3, 2, 2}, // P1
        {9, 0, 2}, // P2
        {2, 2, 2}, // P3
        {4, 3, 3}  // P4
    };

    // Available Resources
    int avail[3] = {3, 3, 2};

    int f[n], ans[n], ind = 0;
    for (k = 0; k < n; k++) {
        f[k] = 0;
    }

    int need[n][m];
    for (i = 0; i < n; i++) {
        for (j = 0; j < m; j++)
            need[i][j] = max[i][j] - alloc[i][j];
    }

    int y = 0;
    for (k = 0; k < n; k++) {
        for (i = 0; i < n; i++) {
            if (f[i] == 0) {
                int flag = 0;
                for (j = 0; j < m; j++) {
                    if (need[i][j] > avail[j]) {
                        flag = 1;
                        break;
                    }
                }
                if (flag == 0) {
                    ans[ind++] = i;
                    for (y = 0; y < m; y++)
                        avail[y] += alloc[i][y];
                    f[i] = 1;
                }
            }
        }
    }

    int flag = 1;
    for (int i = 0; i < n; i++) {
        if (f[i] == 0) {
            flag = 0;
            printf("The following system is not safe\n");
            break;
        }
    }

    if (flag == 1) {
        printf("Following is the SAFE Sequence\n");
        for (i = 0; i < n - 1; i++)
            printf(" P%d ->", ans[i]);
        printf(" P%d", ans[n - 1]);
    }

    return 0;
}
```

Output:

```
karan@karan-VirtualBox:~$ gcc bankers.c -o bankers
karan@karan-VirtualBox:~$ ./bankers
Following is the SAFE Sequence
 P1 -> P3 -> P4 -> P0 -> P2
karan@karan-VirtualBox:~$
```