```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>

// Global variables for shared memory
int *pa, *pn;

// Function for John's turn
void johnTurn(int *n) {
    int b = rand() % 8 + 2; // Generate a random number between 2 and 9
    *pa *= b;
    printf("John multiplied p by %d, resulting p is %d\n", b, *pa);
}

// Function for Michael's turn
void michaelTurn(int *n) {
    int b = rand() % 8 + 2; // Generate a random number between 2 and 9
    *pa *= b;
    printf("Michael multiplied p by %d, resulting p is %d\n", b, *pa);
}

int main() {
    // Get input from the user
    printf("Enter the value of n: ");
    int temp;
    scanf("%d", &temp);
    pn = &temp;
    // Create shared memory for pa
    int shmid = shmget(IPC_PRIVATE, sizeof(int), IPC_CREAT | 0666);
    pa = (int*) shmat(shmid, NULL, 0);
    // Initialize shared memory variables
    *pa = 1;
    // Fork a process
    pid_t pid = fork();
    if (pid < 0) {
        fprintf(stderr, "Fork failed\n");
        return 1;
    } else if (pid == 0) { // Child process (John)
        srand(getpid()); // Seed random number generator for John
        while (*pa <= *pn) {
            johnTurn(pn);
            if (*pa > *pn) {
                printf("John wins!\n");
                break;
            }
            sleep(1);
        }
    } else { // Parent process (Michael)
        srand(getpid()); // Seed random number generator for Michael
        while (*pa <= *pn) {
            michaelTurn(pn);
            if (*pa > *pn) {
                printf("Michael wins!\n");
                break;
            }
            sleep(1);
        }
    }

    shmdt(pa);
    shmctl(shmid, IPC_RMID, NULL);
    return 0;
}
```