```c
#include <stdio.h>
#include <stdlib.h>
#define MAX 5
int mutex = 1;
int wait(int s) {
    while (s <= 0);
    return --s;
}
int signal(int s) {
    return ++s;
}
void eat(int philosopher) {
    printf("P %d is granted to eat\n", philosopher);
    sleep(1);
    printf("P %d has finished eating\n", philosopher);
}
int main() {
    int total = MAX;
    int hungryCount;
    int hungry[MAX] = {0};
    int positions[MAX];
    printf("Enter the total number of philosophers: %d\n", total);
    printf("How many are hungry: ");
    scanf("%d", &hungryCount);
    for (int i = 0; i < hungryCount; i++) {
        printf("Enter philosopher %d position (1 to %d): ", i + 1, total);
        scanf("%d", &positions[i]);
        positions[i]--;
        hungry[positions[i]] = 1;
    }
int choice;
do {
        printf("\n1. One can eat at a time\n2. Two can eat at a time\n3. Exit\nEnter your choice: ");
        scanf("%d", &choice);
        if (choice == 1) {
            printf("Allow one philosopher to eat at any time\n");
            for (int i = 0; i < hungryCount; i++) {
                int p = positions[i] + 1;
                printf("P %d is waiting\n", p);
            }
```

```
    }
int choice;
do {
        printf("\n1. One can eat at a time\n2. Two can eat at a time\n3. Exit\nEnter your choice: ");
        scanf("%d", &choice);
        if (choice == 1) {
            printf("Allow one philosopher to eat at any time\n");
            for (int i = 0; i < hungryCount; i++) {
                int p = positions[i] + 1;
                printf("P %d is waiting\n", p);
            }
            for (int i = 0; i < hungryCount; i++) {
                mutex = wait(mutex);
                eat(positions[i] + 1);
                mutex = signal(mutex);
            }
        } else if (choice == 2) {
            printf("Allow two philosophers to eat at any time\n");
            int i = 0;
            while (i < hungryCount) {
                int count = 0;
                while (count < 2 && i < hungryCount) {
                    printf("P %d is waiting\n", positions[i] + 1);
                    i++;
                    count++;
                }

                for (int j = i - count; j < i; j++) {
                    mutex = wait(mutex);
                    eat(positions[j] + 1);
                    mutex = signal(mutex);
                }
            }
        }

    } while (choice != 3);

    return 0;
}
```

```
Enter the total number of philosophers: 5
How many are hungry: 3
Enter philosopher 1 position (1 to 5): 2
Enter philosopher 2 position (1 to 5): 4
Enter philosopher 3 position (1 to 5): 5

1. One can eat at a time
2. Two can eat at a time
3. Exit
Enter your choice: 1
Allow one philosopher to eat at any time
P 2 is waiting
P 4 is waiting
P 5 is waiting
P 2 is granted to eat
P 2 has finished eating
P 4 is granted to eat
P 4 has finished eating
P 5 is granted to eat
P 5 has finished eating

1. One can eat at a time
2. Two can eat at a time
3. Exit
Enter your choice: 2
Allow two philosophers to eat at any time
P 2 is waiting
P 4 is waiting
P 2 is granted to eat
P 2 has finished eating
P 4 is granted to eat
P 4 has finished eating
P 5 is waiting
P 5 is granted to eat
P 5 has finished eating

1. One can eat at a time
2. Two can eat at a time
3. Exit
Enter your choice: 3

Process returned 0 (0x0)   execution time : 16.160 s
Press any key to continue.
```