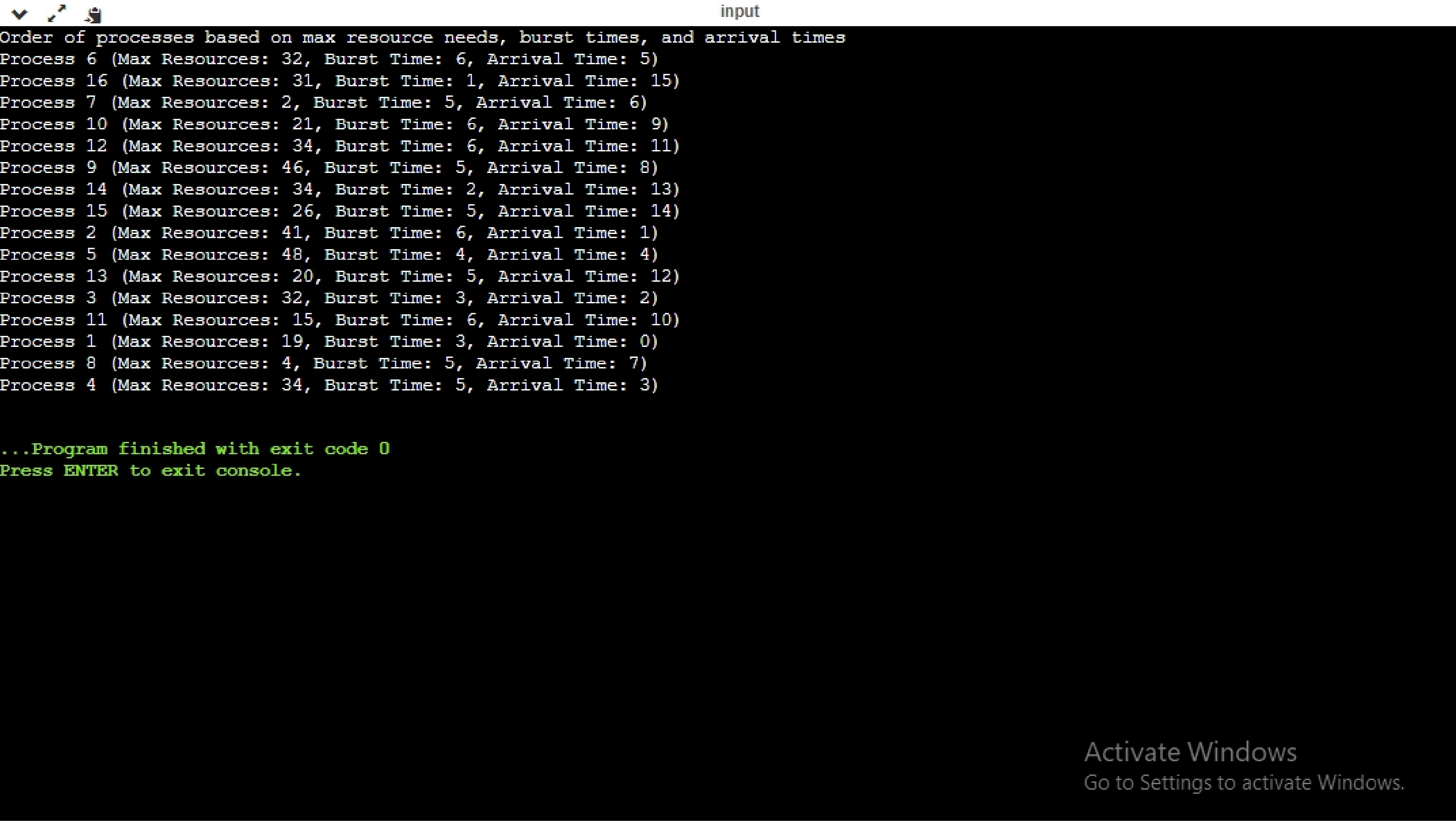
Code for modified scheduling algorithm

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

// Define the number of processes
#define PROCESSES 16

int pid[PROCESSES];
int max_resources[PROCESSES];
int burst_times[PROCESSES] = {0}; //     Initializing the burst time to 0
int arrival_times[PROCESSES] = {0};

// Function to compare processes by max_resources, burst_times, and arrival_times (used for sorting)
int compareProcesses(int a, int b)
{
    if (max_resources[a] == max_resources[b])
    {
        if (burst_times[a] == burst_times[b])
        {
            return arrival_times[a] - arrival_times[b];
        }
        return burst_times[a] - burst_times[b];
    }
    return max_resources[a] - max_resources[b];
}

// Function to perform sorting using bubble sort
```

```
void Sort(int* arr, int n)
{
    int i, j;
    for (i = 0; i < n - 1; i++)
    {
        for (j = 0; j < n - i - 1; j++)
        {
            if (compareProcesses(arr[j], arr[j + 1]) > 0)
            {
                // Swap process IDs
                int temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
        }
    }
}

int main()
{
    int i;
    srand(time(0));

    // Generate random max_resources and burst_times for each process
    for (i = 0; i < PROCESSES; i++) {
        pid[i] = i + 1; // PIDs from 1 to 10
        max_resources[i] = rand() % 50 + 1; // Generates a random number between 1 and 50
        burst_times[i] = rand() % 6 + 1; // Generates a random number between 1 and 6

        arrival_times[i] = i; // Set arrival times from 0 to 9

    }


    // Sort the pid's based on max resource needs, burst times and arrival times using bubble sort

    Sort(pid, PROCESSES);


    printf("Order of processes based on max resource needs, burst times, and arrival times \n");

    for (i = 0; i < PROCESSES; i++)

    {

        int p = pid[i];

        printf("Process %d (Max Resources: %d, Burst Time: %d, Arrival Time: %d)\n", p, max_resources[p - 1], burst_times[p - 1], arrival_times[p - 1]);

    }

    return 0;

}
```

OUTPUT of 16 Processes



```
Order of processes based on max resource needs, burst times, and arrival times
Process 6 (Max Resources: 32, Burst Time: 6, Arrival Time: 5)
Process 16 (Max Resources: 31, Burst Time: 1, Arrival Time: 15)
Process 7 (Max Resources: 2, Burst Time: 5, Arrival Time: 6)
Process 10 (Max Resources: 21, Burst Time: 6, Arrival Time: 9)
Process 12 (Max Resources: 34, Burst Time: 6, Arrival Time: 11)
Process 9 (Max Resources: 46, Burst Time: 5, Arrival Time: 8)
Process 14 (Max Resources: 34, Burst Time: 2, Arrival Time: 13)
Process 15 (Max Resources: 26, Burst Time: 5, Arrival Time: 14)
Process 2 (Max Resources: 41, Burst Time: 6, Arrival Time: 1)
Process 5 (Max Resources: 48, Burst Time: 4, Arrival Time: 4)
Process 13 (Max Resources: 20, Burst Time: 5, Arrival Time: 12)
Process 3 (Max Resources: 32, Burst Time: 3, Arrival Time: 2)
Process 11 (Max Resources: 15, Burst Time: 6, Arrival Time: 10)
Process 1 (Max Resources: 19, Burst Time: 3, Arrival Time: 0)
Process 8 (Max Resources: 4, Burst Time: 5, Arrival Time: 7)
Process 4 (Max Resources: 34, Burst Time: 5, Arrival Time: 3)


...Program finished with exit code 0
Press ENTER to exit console.
```