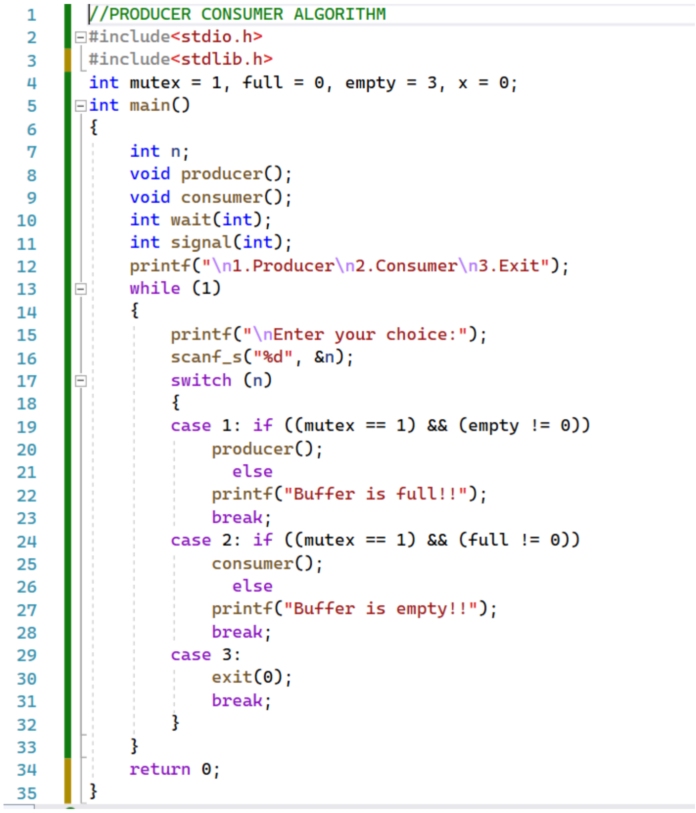
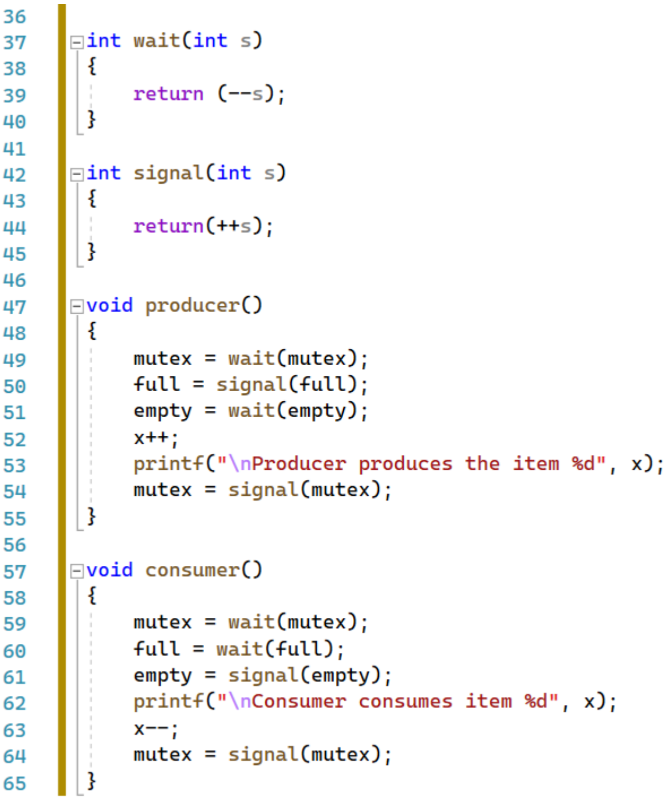
# Operating Systems – Lab#4 SOLUTIONS

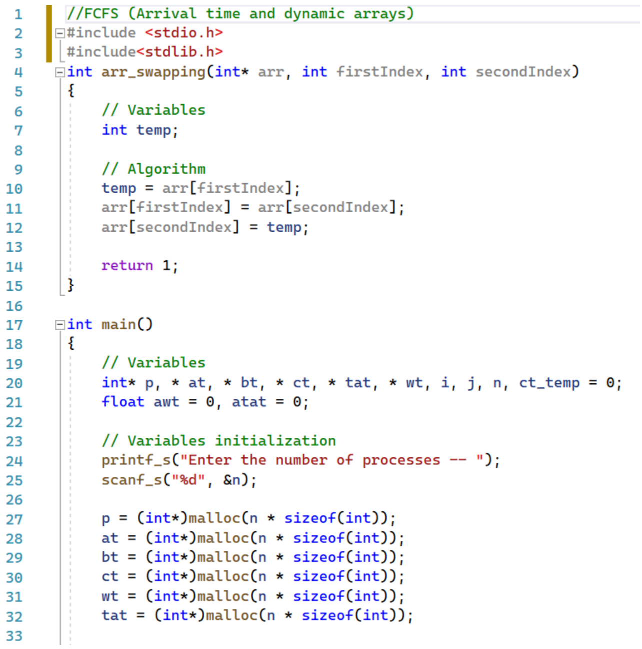


```c
//PRODUCER CONSUMER ALGORITHM
#include<stdio.h>
#include<stdlib.h>
int mutex = 1, full = 0, empty = 3, x = 0;
int main()
{
    int n;
    void producer();
    void consumer();
    int wait(int);
    int signal(int);
    printf("\n1.Producer\n2.Consumer\n3.Exit");
    while (1)
    {
        printf("\nEnter your choice:");
        scanf_s("%d", &n);
        switch (n)
        {
        case 1: if ((mutex == 1) && (empty != 0))
            producer();
              else
            printf("Buffer is full!!");
            break;
        case 2: if ((mutex == 1) && (full != 0))
            consumer();
              else
            printf("Buffer is empty!!");
            break;
        case 3:
            exit(0);
            break;
        }
    }
    return 0;
}
```

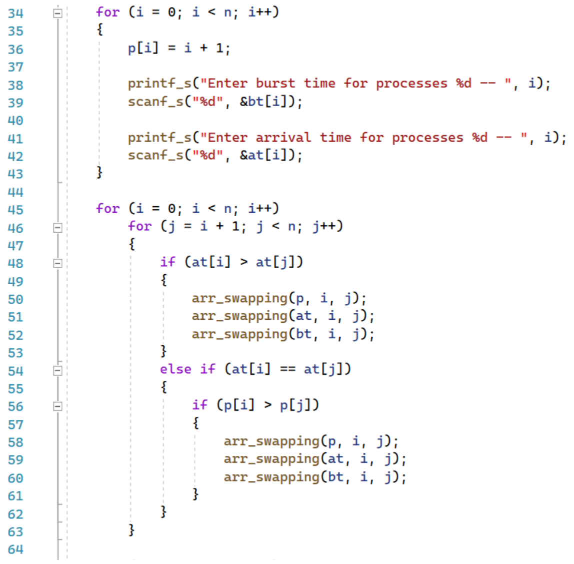


```c

int wait(int s)
{
    return (--s);
}

int signal(int s)
{
    return(++s);
}

void producer()
{
    mutex = wait(mutex);
    full = signal(full);
    empty = wait(empty);
    x++;
    printf("\nProducer produces the item %d", x);
    mutex = signal(mutex);
}

void consumer()
{
    mutex = wait(mutex);
    full = wait(full);
    empty = signal(empty);
    printf("\nConsumer consumes item %d", x);
    x--;
    mutex = signal(mutex);
}
```

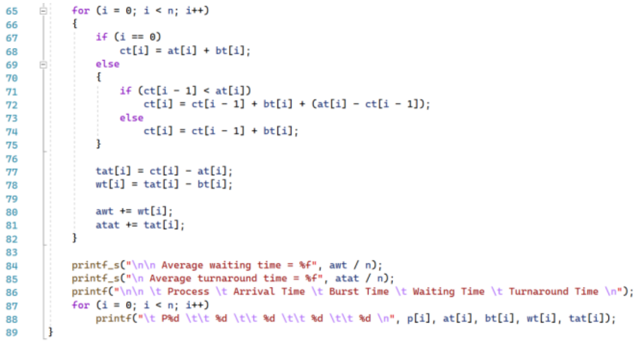
# Operating Systems – Lab#4 SOLUTIONS



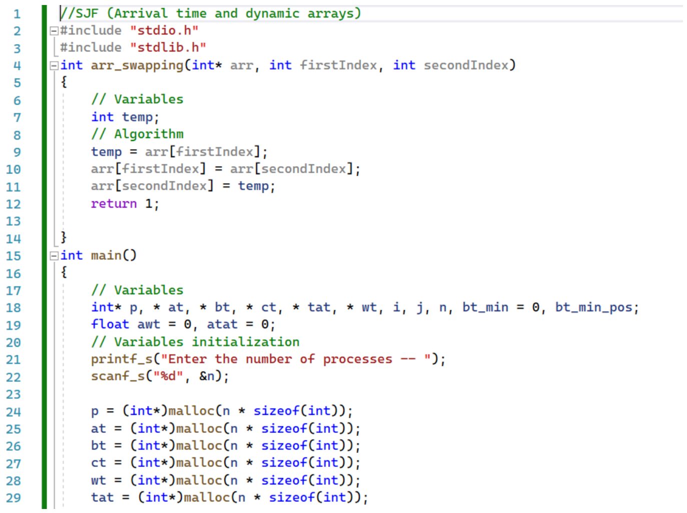
```c
//FCFS (Arrival time and dynamic arrays)
#include <stdio.h>
#include<stdlib.h>
int arr_swapping(int* arr, int firstIndex, int secondIndex)
{
    // Variables
    int temp;

    // Algorithm
    temp = arr[firstIndex];
    arr[firstIndex] = arr[secondIndex];
    arr[secondIndex] = temp;

    return 1;
}

int main()
{
    // Variables
    int* p, * at, * bt, * ct, * tat, * wt, i, j, n, ct_temp = 0;
    float awt = 0, atat = 0;

    // Variables initialization
    printf_s("Enter the number of processes -- ");
    scanf_s("%d", &n);

    p = (int*)malloc(n * sizeof(int));
    at = (int*)malloc(n * sizeof(int));
    bt = (int*)malloc(n * sizeof(int));
    ct = (int*)malloc(n * sizeof(int));
    wt = (int*)malloc(n * sizeof(int));
    tat = (int*)malloc(n * sizeof(int));

```



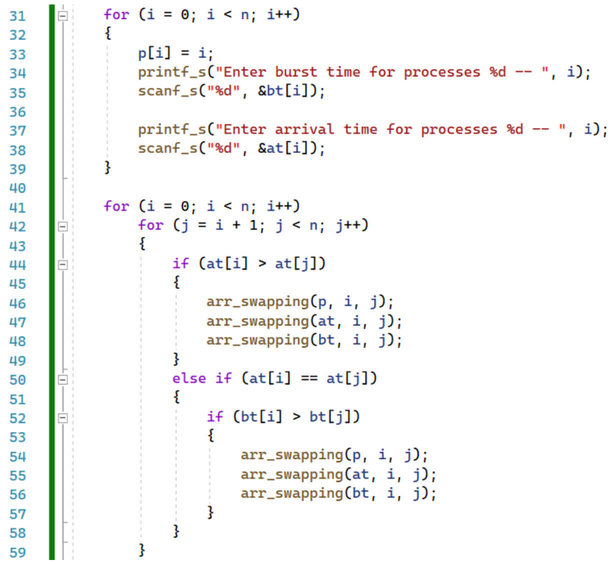
```c
    for (i = 0; i < n; i++)
    {
        p[i] = i + 1;

        printf_s("Enter burst time for processes %d -- ", i);
        scanf_s("%d", &bt[i]);

        printf_s("Enter arrival time for processes %d -- ", i);
        scanf_s("%d", &at[i]);
    }

    for (i = 0; i < n; i++)
        for (j = i + 1; j < n; j++)
        {
            if (at[i] > at[j])
            {
                arr_swapping(p, i, j);
                arr_swapping(at, i, j);
                arr_swapping(bt, i, j);
            }
            else if (at[i] == at[j])
            {
                if (p[i] > p[j])
                {
                    arr_swapping(p, i, j);
                    arr_swapping(at, i, j);
                    arr_swapping(bt, i, j);
                }
            }
        }

```



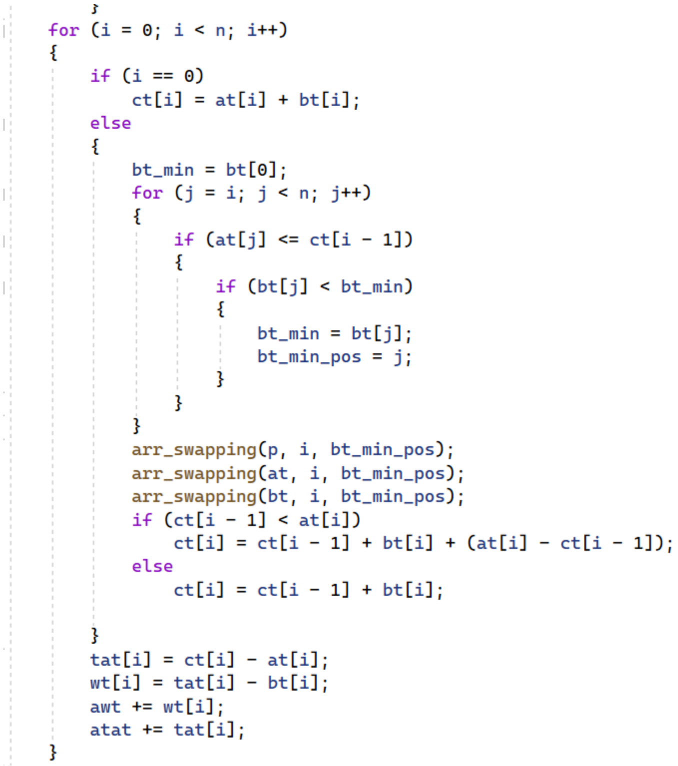
```c
    for (i = 0; i < n; i++)
    {
        if (i == 0)
            ct[i] = at[i] + bt[i];
        else
        {
            if (ct[i - 1] < at[i])
                ct[i] = ct[i - 1] + bt[i] + (at[i] - ct[i - 1]);
            else
                ct[i] = ct[i - 1] + bt[i];
        }

        tat[i] = ct[i] - at[i];
        wt[i] = tat[i] - bt[i];

        awt += wt[i];
        atat += tat[i];
    }

    printf_s("\n\n Average waiting time = %f", awt / n);
    printf_s("\n Average turnaround time = %f", atat / n);
    printf("\n\n \t Process \t Arrival Time \t Burst Time \t Waiting Time \t Turnaround Time \n");
    for (i = 0; i < n; i++)
        printf("\t P%d \t\t %d \t\t %d \t\t %d \t\t %d \n", p[i], at[i], bt[i], wt[i], tat[i]);
}
```



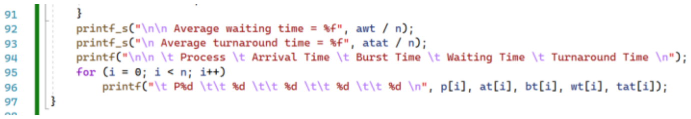
```c
//SJF (Arrival time and dynamic arrays)
#include "stdio.h"
#include "stdlib.h"
int arr_swapping(int* arr, int firstIndex, int secondIndex)
{
    // Variables
    int temp;
    // Algorithm
    temp = arr[firstIndex];
    arr[firstIndex] = arr[secondIndex];
    arr[secondIndex] = temp;
    return 1;

}
int main()
{
    // Variables
    int* p, * at, * bt, * ct, * tat, * wt, i, j, n, bt_min = 0, bt_min_pos;
    float awt = 0, atat = 0;
    // Variables initialization
    printf_s("Enter the number of processes -- ");
    scanf_s("%d", &n);

    p = (int*)malloc(n * sizeof(int));
    at = (int*)malloc(n * sizeof(int));
    bt = (int*)malloc(n * sizeof(int));
    ct = (int*)malloc(n * sizeof(int));
    wt = (int*)malloc(n * sizeof(int));
    tat = (int*)malloc(n * sizeof(int));
```



```c
    for (i = 0; i < n; i++)
    {
        p[i] = i;
        printf_s("Enter burst time for processes %d -- ", i);
        scanf_s("%d", &bt[i]);

        printf_s("Enter arrival time for processes %d -- ", i);
        scanf_s("%d", &at[i]);
    }

    for (i = 0; i < n; i++)
        for (j = i + 1; j < n; j++)
        {
            if (at[i] > at[j])
            {
                arr_swapping(p, i, j);
                arr_swapping(at, i, j);
                arr_swapping(bt, i, j);
            }
            else if (at[i] == at[j])
            {
                if (bt[i] > bt[j])
                {
                    arr_swapping(p, i, j);
                    arr_swapping(at, i, j);
                    arr_swapping(bt, i, j);
                }
            }
        }
```



```
    }
for (i = 0; i < n; i++)
{
    if (i == 0)
        ct[i] = at[i] + bt[i];
    else
    {
        bt_min = bt[0];
        for (j = i; j < n; j++)
        {
            if (at[j] <= ct[i - 1])
            {
                if (bt[j] < bt_min)
                {
                    bt_min = bt[j];
                    bt_min_pos = j;
                }
            }
        }
        arr_swapping(p, i, bt_min_pos);
        arr_swapping(at, i, bt_min_pos);
        arr_swapping(bt, i, bt_min_pos);
        if (ct[i - 1] < at[i])
            ct[i] = ct[i - 1] + bt[i] + (at[i] - ct[i - 1]);
        else
            ct[i] = ct[i - 1] + bt[i];

    }
    tat[i] = ct[i] - at[i];
    wt[i] = tat[i] - bt[i];
    awt += wt[i];
    atat += tat[i];
}
```



```
    }
    printf_s("\n\n Average waiting time = %f", awt / n);
    printf_s("\n Average turnaround time = %f", atat / n);
    printf("\n\n \t Process \t Arrival Time \t Burst Time \t Waiting Time \t Turnaround Time \n");
    for (i = 0; i < n; i++)
        printf("\t P%d \t\t %d \t\t %d \t\t %d \t\t %d \n", p[i], at[i], bt[i], wt[i], tat[i]);
}
```