구현 과정

입력과 시뮬레이팅 과정 구현

먼저 큐 자료구조가 필요하다고 생각해서, 배열로 구현하려고 했습니다. 그러나 Shortest Job First 스케줄링을 생각해봤을 때 중간 원소의 삭제가 불편한 점이 있어 배열 대신 연결 리스트로 만들었습니다. 그래서 프로세스 구조체를 다음과 같이 정의했습니다.

```
typedef struct Process
{
    int pid;
    int priority;
    int arrivalTime;
    int serviceTime;
    int elapsedTime;
    int state;
    int ticket;
    struct Process* next;
    struct Process* prev;
} Process;
```

```
enum STATE
{
    STATE_NOT_ARRIVED,
    STATE_ARRIVED,
    STATE_READY,
    STATE_RUNNING,
    STATE_TERMINATED
};
```

그리고 프로세스의 상태를 나타내는 state는 enum을 사용해 정의했습니다.

이러한 프로세스 구조체를 PROCESS_COUNT(#define된 상수)만큼 갖고 있는 processArray를 정의해서 시뮬레이팅하기 위한 워크로드 정보를 담기 위한 배열로 사용합니다. 프로그램이 실행되면 main함수에서 ProcessInput() 함수를 통해 프로세스 정보를 입력받습니다. 그 다음 CreateProcess()로 프로세스를 생성해서 processArray에 넣습니다.

```
// Processing user input
void ProcessInput()
{
    int i;
    int arrivalTime, serviceTime, ticket;
    for (i = 0; i < PROCESS_COUNT; i++)
    {
        printf("Please enter the Arrival time and Service time of process %c: ", i + 'A');
        scanf("%d %d", &arrivalTime, &serviceTime);
        printf("Please enter the Ticket of the process %c: ", i + 'A');
        scanf("%d", &ticket);
        if (serviceTime == 0 || serviceTime < 0)
        {
            // Invalid input handling
            printf("Service time should not be zero or less than zero. Please enter again.\n");
            i--;
            continue;
        }
        else if (arrivalTime > TOTAL_TIME || arrivalTime < 0)
        {
            // Invalid input handling
            printf("Arrival time should not be greater than %d or less than zero. Please enter again.\n", TOTAL_TIME);
            i--;
            continue;
        }
        else
        {
            // if input is valid, create process
            Process* createdProcess = CreateProcess(i+'A', arrivalTime, serviceTime, ticket);
            // Add created process to array
            processArray[i] = createdProcess;
        }
    }
}
```

ProcessInput함수입니다. 잘못된 정보가 입력되면 다시 입력 받고 올바른 정보가 입력되면 프로세스를 생성해 배열에 넣습니다.

```
Process* CreateProcess(int pid, int arrivalTime, int serviceTime, int ticket)
{
    // Create process
    Process* newProcess = (Process*)malloc(sizeof(struct Process));
    newProcess->pid = pid;
    newProcess->priority = 0;
    newProcess->arrivalTime = arrivalTime;
    newProcess->serviceTime = serviceTime;
    newProcess->elapsedTime = 0;
    newProcess->state = STATE_NOT_ARRIVED;
    newProcess->next = NULL;
    newProcess->prev = NULL;
    newProcess->ticket = ticket;
    return newProcess;
```

CreateProcess함수입니다. 입력받은 정보를 바탕으로 NOT_ARRIVED 상태인 프로세스를 생성합니다. 정보 입력과 프로세스 생성이 끝나면 Simuate()를 통해 스케줄링을 시뮬레이트합니다.

```
void Simulate(int method)
{
    // Simulate process scheduling
    int time = 0;
    Initialize(method);
    while (time <= TOTAL_TIME)
    {
        ProcessArrive(time); // When process arrives at time, add it to list
        Schedule(method); // Using scheduling methods, Pick one process to execute
        if (runningProcess != NULL)
        {
            // process execution simulate
            runningProcess->state = STATE_RUNNING;
            runningProcess->elapsedTime++;
            timeSpent++;
            executionHistory[time] = runningProcess->pid;
            if (runningProcess->elapsedTime >= runningProcess->serviceTime)
            {
                // process's work complete
                runningProcess->state = STATE_TERMINATED;
                runningProcess = NULL;
                timeSpent = 0;
            }
        }
        else
        {
            // There are no processes to run, CPU goes idle
            executionHistory[time] = '.';
        }
        time++;
    }
    PrintHistory();
}
```

0초부터 시작해 TOTAL_TIME(#define된 상수)까지 스케줄링을 시뮬레이트합니다. 이 함수에서는 먼저 ProcessArrive(time)을 실행합니다. ProcessArrive(time)은 시뮬레이팅 시간이 time이 됐을 때 NOT_ARRIVED상태인 프로세스들을 ARRIVED상태로 바꾸고 프로세스 리스트에 추가하는 역할을 합니다. 프로세스 리스트 정의는 다음과 같습니다.

```
// Because of MLFQ scheduling, made array of linked list of process
// 0 is used in normal scheduling method
// in MLFQ scheduling method, processList[0]=Q0, [1]=Q1, [2]=Q2... / timeQuantum[0]=q^0, [1]=q^1, [2]=q^2 ...
// priority order: (high) processList[0] > processList[1] > processList[2] > ... (low)
Process* processList[NUMBER_OF_MLFQ];
```

처음에는 단일 리스트였지만 Multi Level Feedback Queue를 구현하면서 리스트의 배열로 바꿨습니다. 멀티 큐가 필요 없는 스케줄링 방법에선 processList[0]만을 사용하고, MLFQ에선 배열의 리스트를 전부 다 사용합니다. 우선순위는 [0]이 가장 높고 점점 낮아지는 내림차순이며, Time quantum은 [0]이 가장 작고 점점 높아지는 오름차순입니다. 이 리스트와 관련된 함수로는 큐의 가장 뒤쪽에 프로세스를 추가하는 PushBack, 가장 앞의 프로세스를 제거하는 Pop, pid로 인덱스를 찾는 FindIndexById, 인덱스로 프로세스를 제거하는 RemoveAt, 두 함수를 바탕으로 pid로 제거하는 Remove()가 있습니다. 처음에는 단일 리스트에만 작동하게 구현했었지만 마찬가지로 MLFQ를 구현하면서 매개변수로 processList의 인덱스를 넘겨주어 해당 processList[index]에 연산을 하게끔 변경했습니다. MLFQ가 아닌 다른 스케줄링 방법일 경우 매개변수로 0을 넘겨주어 processList[0]만을 사용합니다.

ProcessArrive가 실행된 다음에는 Schedule 함수를 실행합니다. 이 함수는 매개변수 method에 따라 각각의 스케줄링 함수를 실행하는 역할을 합니다. method는 enum으로 정의되어 있습니다. Schedule함수가 실행된 후에는 변수 runningProcess에 새로 실행할 프로세스가 할당됩니다.

이후 runningProcess가 NULL이 아닐 경우(실행할 프로세스가 있을 경우) 해당 프로세스의 CPU를 할당 받은 시간 변수인 elapsedTime을 증가시킵니다. 또 상태를 RUNNING으로 바꿔주며 나중에 출력을 위해 실행된 프로세스를 기록해둡니다. runningProcess가 NULL일 경우에는 큐에 프로세스가 없는 경우(아직 도착하지 않았거나 이미 작업을 다 처리한 경우 등)이기 때문에 특별한 작업을 하지 않고 출력을 위해 기록만 해둡니다. 다음으로 Time quantum에 따라서도 스케줄링을 해야 하기 때문에 이를 위한

변수인 timeSpent도 증가시켜줍니다. 만약 serviceTime만큼 실행했다면 작업이 다 완료되었다는 뜻이므로 상태를 TERMINATED로 바꿔줍니다.

여기까지가 시뮬레이터의 한 사이클이며 다 완료되면 time을 증가시켜 TOTAL_TIME까지시뮬레이팅을 반복합니다. TOTAL_TIME까지도 모두 실행이 끝나면 스케줄링되어 실행했던 프로세스들을 전부 출력합니다.

스케줄링 구현

FIFO (First In First Out)

```
void FirstInFirstOut()
{
    if (runningProcess != NULL)
    {
        // Yield to runningProcess to execute
        return;
    }
    else
    {
        // Processes in processQueue are ordered by arrival time
        // so the first element is "first In" process, execute first
        runningProcess = Pop(0);
    }
}
```

도착 시간이 빠른 순서대로 processList에 넣어지므로, 이 리스트(큐)의 맨 앞 프로세스를 꺼내서 runningProcess에 할당합니다. 따라서 먼저 도착한 프로세스가 먼저 실행되는 First In First Out방식으로 실행됩니다. 현재 실행중인 프로세스가 이미 있을 경우에는 비-선점방식이기 때문에 그 프로세스를 그대로 실행합니다.(runningProcess를 바꾸지 않고 종료할 경우 Simulate함수에서 계속 실행되는 것 처럼 처리됩니다.)

Round Robin(q = 1, q = 4)

```
void RoundRobin(int quantum)
{
    if (runningProcess != NULL)
    {
        if (timeSpent >= quantum)
        {
            // timeSpent of runningProcess >= time quantum. Push it back to the list.
            timeSpent = 0;
            PushBack(runningProcess, 0);
            runningProcess = Pop(0);
        }
        else
        {
            // need more time to current runningProcess. Yield to runningProcess
            return;
        }
    }
    else
    {
        runningProcess = Pop(0);
    }
}
```

현재 실행중인 프로세스가 있을 경우에는, 그 프로세스가 사용한 시간이 time quantum보다 클 때 다른 프로세스를 실행합니다(선점 방식). 이때 실행 중이던 프로세스는 큐에 다시 넣고, 다른 프로세스는 큐에서 꺼내서 실행합니다. 현재 실행중인 프로세스가 없을 경우에는, 단순하게 큐에서 꺼내서 실행합니다. q=1일때와 q=4일때를 각각 구현하는 것보다 매개변수로 time quantum을 받아 RoundRobin(1), RoundRobin(4)를 호출하는 방식으로 구현했습니다.

Shortest Job First

```
void ShortestJobFirst()
{
    int shortestTime = 2147483647;
    Process* iterator = processList[0];
    Process* shortest = NULL;
    // Find Shortest Job in process list
    while (iterator != NULL)
    {
        if (iterator->serviceTime < shortestTime)
        {
            shortestTime = iterator->serviceTime;
            shortest = iterator;
        }
        else if (iterator->serviceTime == shortestTime)
        {
            if (shortest->pid >= iterator->pid)
            {
                shortestTime = iterator->serviceTime;
                shortest = iterator;
            }
        }
        iterator = iterator->next;
    }
    if (runningProcess != NULL)
    {
        // If there are already runningProcess, Yield
        return;
    }
    else
    {
        if (shortest != NULL)
        {
            // Run shortest Job
            runningProcess = Remove(shortest->pid, 0);
        }
    }
}
```

현재 리스트(큐)에 있는 프로세스들 중 가장 service time이 작은 프로세스를 찾아 실행합니다. 실행하고 있는 프로세스가 있을 경우엔 비-선점 방식이므로 양보합니다.

Multi Level Feedback Queue (q=1, q=2^i)

```
void MultiLevelFeedbackQueue()
{
    // Priority order 0   >  1  >  2 ...(low number is high priority)
    // Time Quantum   q^0 > q^1 > q^2...
    int i;
    // From high priority queue to low priority
    // Find highest priority process (front node)
    Process* highest = NULL;
    for (i = 0; i < NUMBER_OF_MLFQ && highest == NULL; i++)
    {
        highest = processList[i];
    }
    if (highest != NULL)
    {
        if (runningProcess != NULL)
        {
            int newPriority = (runningProcess->priority + 1 >= NUMBER_OF_MLFQ) ? NUMBER_OF_MLFQ-1 : runningProcess->priority + 1;
            if (timeSpent >= timeQuantum[runningProcess->priority])
            {
                // If runningProcess's serviceTime > time quantum of Multi Level Feedback Queue
                timeSpent = 0;
                runningProcess->priority = newPriority;
                PushBack(runningProcess, newPriority);
                // Context switch to higher priority process
                runningProcess = Remove(highest->pid, highest->priority);
            }
            else if(timeSpent < timeQuantum[runningProcess->priority])
            {
                // If time slice of current runningProcess is less than timeQuantum, Yield
                return;
            }
        }
        else
        {
            // If there are no runningProcess, execute higher priority process
            runningProcess = Remove(highest->pid, highest->priority);
        }
    }
}
```

우선순위가 높은 큐부터 낮은 큐까지 검사해서, 우선순위가 높은 큐의 첫번째로 넣어진 프로세스를 가져옵니다. 그러고나서 현재 프로세스가 time quantum만큼 시간을 사용했는지 확인합니다. 사용했다면 우선순위를 한단계 낮추고 새로 가져온 프로세스를 실행합니다. 사용하지 않았다면 계속 실행되게 양보합니다. 새로 실행할 프로세스는 우선순위가 가장 높은 것을 실행하고, 현재 실행하고 있는 프로세스는 Time quantum만큼의 실행시간(우선순위가 낮을수록 더 깁니다)을 보장하는 비-선점방식입니다.

실행중인 프로세스가 time quantum을 다 사용하지 않았어도 새로 우선순위가 높은 프로세스가 도착하면 그것을 실행하는 선점 방식도 있습니다. 하지만 이렇게 하면 우선순위가 낮은 프로세스는 계속 프로세스가 도착할 경우에 실행될 수 있는 시간이 비교적 적고, 점점 낮은 우선순위의 큐에 쌓일 것 이라고 생각해서 선점 방식으로 구현해보았습니다.

Lottery

```
void Lottery()
{
    if (processList[0] == NULL)
    {
        return;
    }
    int counter = 0, totalTickets = 0, winner = 0;
    // Calculate total number of tickets
    Process* iterator = processList[0];
    while (iterator != NULL)
    {
        totalTickets += iterator->ticket;
        iterator = iterator->next;
    }
    // Get a value between 0 and the total # of tickets
    winner = rand() % totalTickets;
    // Loop until the sum of ticket values is greater than winner
    iterator = processList[0];
    while (iterator != NULL)
    {
        counter += iterator->ticket;
        if (counter > winner)
        {
            // found the winner
            break;
        }
        iterator = iterator->next;
    }
    // 'iterator' is the winner: schedule it
    if (runningProcess != NULL)
        PushBack(runningProcess, 0);
    runningProcess = Remove(iterator->pid, 0);
}
```

Lottery는 큐에 들어가있는 대기중인 프로세스들의 티켓 수를 합쳐서 `totalTickets`에 저장한 후에 (0, `totalTickets`) 범위의 수를 하나 뽑아 `winner`에 저장합니다. 이후 프로세스 수만큼 순회를 하며 프로세스의 티켓만큼 계속 더해갑니다. 이 더한 값들이 순회 중 `winner`보다 커질 경우 해당 프로세스가 당첨되어 실행됩니다. 이미 실행중인 프로세스가 있을 경우 해당 프로세스를 큐에 넣고, 당첨된 프로세스를 큐에서 꺼내어 우선적으로 실행하는 선점 방식입니다.

어려웠던 점

전체적인 실행 구조와 프로그램 출력방식에 대해 고민했습니다. 그림과 같은 결과를 출력하기 위해서 초마다 프로세스가 어떻게 스케줄링되고 실행되는지 의사코드를 만들어 써보고 각각의 함수를 차근차근 만들었습니다. 그러다가 처음엔 배열로 만들었던 프로세스 큐를 SJF나 MLFQ를 구현하면서 리스트로 바꿀 필요성을 느껴 리팩토링하는데에 시간이 꽤 걸렸습니다. 마지막으로 MLFQ의 규칙과 구현방법에 대한 고민(선점, 비-선점 등등)을 하다가 결국 교재ppt와 같은 출력이 나오는 방법을 택했습니다.

실행 결과

24p workload

```
Please enter the Arrival time and Service time of process A: 0 3
Please enter the Ticket of the process A: 100
Please enter the Arrival time and Service time of process B: 2 6
Please enter the Ticket of the process B: 100
Please enter the Arrival time and Service time of process C: 4 4
Please enter the Ticket of the process C: 100
Please enter the Arrival time and Service time of process D: 6 5
Please enter the Ticket of the process D: 100
Please enter the Arrival time and Service time of process E: 8 2
Please enter the Ticket of the process E: 100

--------  First In First Out-----------------
TIME: 0 to 20
A AAA_________________
B ___BBBBBB__________
C _________CCCC______
D _____________DDDDD__
E ___________________EE

--------  Round Robin q=1 -------------------
TIME: 0 to 20
A AA_A________________
B __B_B_B__B__B__B__
C _____C__C__C__C___
D _______D___D___D__DD
E ___________E___E_____

--------  Round Robin q=4 -------------------
TIME: 0 to 20
A AAA_________________
B ___BBBB________BB___
C _______CCCC_________
D ____________DDDD____D
E ___________________EE_

--------  Shortest Job First ----------------
TIME: 0 to 20
A AAA_________________
B ___BBBBBB__________
C ____________CCCC____
D ________________DDDDD
E _________EE_________

------- Mutli Level Feeback Queue q=1 -------
TIME: 0 to 20
A AA_A________________
B __B__B_____B__B__B_B
C ____C__C____C__C____
D ______D__D___D__D_D_
E _________E_E_________

------- Mutli Level Feeback Queue q=4 -------
TIME: 0 to 20
A AA_A________________
B __B__BB_______BBB___
C ____C____CC______C__
D _______D___DD_____DD
E _________E____E______

---------------- LOTTERY --------------------
TIME: 0 to 20
A AA_A________________
B __B__B_B__B_____B_B
C ____C_C_C_______C___
D __________D_D_D_D__D_
E ____________E_E______
```

Process	Arrival Time	Service Time
A	0	3
B	2	6
C	4	4
D	6	5
E	8	2

Custom workload #1 – 프로세스가 동시에 도착했을 때

```
Please enter the Arrival time and Service time of process A: 0 4
Please enter the Ticket of the process A: 100
Please enter the Arrival time and Service time of process B: 0 4
Please enter the Ticket of the process B: 100
Please enter the Arrival time and Service time of process C: 0 4
Please enter the Ticket of the process C: 100
Please enter the Arrival time and Service time of process D: 0 4
Please enter the Ticket of the process D: 100
Please enter the Arrival time and Service time of process E: 0 4
Please enter the Ticket of the process E: 100

--------  First In First Out-----------------
TIME: 0 to 20
A AAAA________________
B ____BBBB____________
C ________CCCC________
D ____________DDDD____
E ________________EEEE

--------  Round Robin q=1 -------------------
TIME: 0 to 20
A A___A___A___A___A___
B _B___B___B___B___B__
C __C___C___C___C___C_
D ___D___D___D___D___D
E ____E___E___E___E___

--------  Round Robin q=4 -------------------
TIME: 0 to 20
A AAAA________________
B ____BBBB____________
C ________CCCC________
D ____________DDDD____
E ________________EEEE

--------  Shortest Job First ----------------
TIME: 0 to 20
A AAAA________________
B ____BBBB____________
C ________CCCC________
D ____________DDDD____
E ________________EEEE

-------  Mutli Level Feeback Queue q=1 -------
TIME: 0 to 20
A A___A___A___A___A___
B _B___B___B___B___B__
C __C___C___C___C___C_
D ___D___D___D___D___D
E ____E___E___E___E___

-------  Mutli Level Feeback Queue q=4 -------
TIME: 0 to 20
A A___AA_______A______
B _B____BB_____B_____B
C __C_____CC_____C___C
D ___D_____DD____D___D
E ____E_____EE____E___

----------------  LOTTERY --------------------
TIME: 0 to 20
A __A_A_A_______A_____
B _____B___B_B_B_____B
C _______C_C__C___C___
D D_____________D_DD__
E _E_E___E___E________
```

Process	Arrival Time	Service Time
A	0	4
B	0	4
C	0	4
D	0	4
E	0	4

Custom workload #2 – 처음 도착한 프로세스의 service time이 매우 길 때

```
Please enter the Arrival time and Service time of process A: 0 12
Please enter the Ticket of the process A: 100
Please enter the Arrival time and Service time of process B: 2 2
Please enter the Ticket of the process B: 100
Please enter the Arrival time and Service time of process C: 4 2
Please enter the Ticket of the process C: 100
Please enter the Arrival time and Service time of process D: 6 2
Please enter the Ticket of the process D: 100
Please enter the Arrival time and Service time of process E: 8 2
Please enter the Ticket of the process E: 100

--------  First In First Out-----------------
TIME: 0 to 20
A AAAAAAAAAAAA________
B ____________BB______
C ______________CC____
D ________________DD__
E __________________EE

--------  Round Robin q=1 -------------------
TIME: 0 to 20
A AA_A__A__A__A_AAAAAA
B __B_B_______________
C _____C__C___________
D _______D___D________
E _________E__E_______

--------  Round Robin q=4 -------------------
TIME: 0 to 20
A AAAA____AAAA____AAAA
B ____BB______________
C ______CC____________
D ____________DD______
E ______________EE____

--------  Shortest Job First ----------------
TIME: 0 to 20
A AAAAAAAAAAAA________
B ____________BB______
C ______________CC____
D ________________DD__
E __________________EE

------- Mutli Level Feeback Queue q=1 -------
TIME: 0 to 20
A AA_A_______AAAAAAAAA
B __B__B______________
C ____C__C____________
D ______D__D__________
E ________E_E_________

------- Mutli Level Feeback Queue q=4 -------
TIME: 0 to 20
A AA_AA_______AAAAAAAA
B __B____B____________
C _____C___C__________
D ______D___D_________
E ________E__E________

---------------- LOTTERY --------------------
TIME: 0 to 20
A AA_A_A__A____AAAAAAA
B __B_B_______________
C _______C__C_________
D ______D____D________
E _________E__E_______
```

Process	Arrival Time	Service Time
A	0	12
B	2	2
C	4	2
D	6	2
E	8	2

Custom workload #3 – 중간에 도착한 프로세스의 service time이 매우 길 때

```
Please enter the Arrival time and Service time of process A: 0 2
Please enter the Ticket of the process A: 100
Please enter the Arrival time and Service time of process B: 2 2
Please enter the Ticket of the process B: 100
Please enter the Arrival time and Service time of process C: 4 12
Please enter the Ticket of the process C: 100
Please enter the Arrival time and Service time of process D: 6 2
Please enter the Ticket of the process D: 100
Please enter the Arrival time and Service time of process E: 8 2
Please enter the Ticket of the process E: 100

--------  First In First Out----------------
TIME: 0 to 20
A AA__________________
B __BB________________
C ____CCCCCCCCCCCC____
D ________________DD__
E __________________EE

--------  Round Robin q=1 -------------------
TIME: 0 to 20
A AA__________________
B __BB________________
C ____CC_C__C_CCCCCCCC
D ______D_D___________
E _________E_E________

--------  Round Robin q=4 -------------------
TIME: 0 to 20
A AA__________________
B __BB________________
C ____CCCC____CCCCCCCC
D _________DD_________
E ___________EE_______

--------  Shortest Job First ----------------
TIME: 0 to 20
A AA__________________
B __BB________________
C ____CCCCCCCCCCCC____
D ________________DD__
E __________________EE

------- Mutli Level Feeback Queue q=1 -------
TIME: 0 to 20
A AA__________________
B __BB________________
C ____CC_C___CCCCCCCCC
D ______D_D___________
E ________E_E_________

------- Mutli Level Feeback Queue q=4 -------
TIME: 0 to 20
A AA__________________
B __BB________________
C ____CC_CC___CCCCCCCC
D ______D___D_________
E _________E_E________

---------------- LOTTERY --------------------
TIME: 0 to 20
A AA__________________
B __BB________________
C ____CC_C__C_CCCCCCCC
D ______D_D___________
E ________E__E________
```

Process	Arrival Time	Service Time
A	0	2
B	2	2
C	4	12
D	6	2
E	8	2

Custom workload #4 –프로세스의 ticket들이 다를 때(Lottery 테스트)

```
Please enter the Arrival time and Service time of process A: 0 4
Please enter the Ticket of the process A: 10
Please enter the Arrival time and Service time of process B: 0 4
Please enter the Ticket of the process B: 50
Please enter the Arrival time and Service time of process C: 0 4
Please enter the Ticket of the process C: 500
Please enter the Arrival time and Service time of process D: 0 4
Please enter the Ticket of the process D: 150
Please enter the Arrival time and Service time of process E: 0 4
Please enter the Ticket of the process E: 30

--------  First In First Out-----------------
TIME: 0 to 20
A AAAA________________
B ____BBBB____________
C ________CCCC________
D ____________DDDD____
E ________________EEEE

--------  Round Robin q=1 -------------------
TIME: 0 to 20
A A___A___A___A___
B _B___B___B___B___
C __C___C___C___C__
D ___D___D___D___D_
E ____E___E___E___E

--------  Round Robin q=4 -------------------
TIME: 0 to 20
A AAAA________________
B ____BBBB____________
C ________CCCC________
D ____________DDDD____
E ________________EEEE

--------  Shortest Job First ----------------
TIME: 0 to 20
A AAAA________________
B ____BBBB____________
C ________CCCC________
D ____________DDDD____
E ________________EEEE

------- Mutli Level Feeback Queue q=1 -------
TIME: 0 to 20
A A___A___A___A___
B _B___B___B___B___
C __C___C___C___C__
D ___D___D___D___D_
E ____E___E___E___E

------- Mutli Level Feeback Queue q=4 -------
TIME: 0 to 20
A A___AA______A___
B _B____BB_____B___
C __C_____CC_____C__
D ___D_____DD____D_
E ____E_____EE____E

---------------- LOTTERY --------------------
TIME: 0 to 20
A _____________A_A_AA
B _______B____B_B_B__
C C_C_C_C_____________
D _D__D__D_D__________
E ___E___E_E_E________
```

Process	Arrival Time	Service Time	Ticket
A	0	4	10
B	0	4	50
C	0	4	500
D	0	4	150
E	0	4	30