NAME: AKHIL

Reg No: 11703357

E-mail: akhildhiman141@gmail.com

Git Hub: https://github.com/akhildhiman7/Student-Teacher-Problem.git

INDEX

S. No	Title	Pg. No
1.	Code	3
2.	Problem in terms of Operating System.	17
3.	Algorithm	17
4.	Complexity	18
5.	Constraint Limit	20
6.	Explanation of Algorithm	29
7.	Condition	30
8.	Test Case	30

<u>CODE:</u> (Python Implementation)

```
import random

class StudentQueue:
    def __init__(self):
        self.items = []
    def isEmpty(self):
        return self.items == []
    def enqueue(self, item, AT, BT):
        self.lst = []
        self.lst.append(item)
        self.lst.append(AT)
        self.lst.append(BT)
        self.items.insert(0,self.lst)
    def dequeue(self):
        return self.items.pop()
    def size(self):
        return len(self.items)
    def head(self):
        return self.items[-1][1]
    def burst_time(self):
        return self.items[-1][2]
    def id_no(self):
        return self.items[-1][0]


class TeacherQueue:
    def __init__(self):
        self.items = []
    def isEmpty(self):
        return self.items == []
    def enqueue(self, item, AT, BT):
        self.lst = []
        self.lst.append(item)
        self.lst.append(AT)
        self.lst.append(BT)
        self.items.insert(0,self.lst)
    def dequeue(self):
        return self.items.pop()
    def size(self):
        return len(self.items)
    def head(self):
        return self.items[-1][1]
    def burst_time(self):
        return self.items[-1][2]
    def id_no(self):
        return self.items[-1][0]



SQ = StudentQueue()
TQ =  TeacherQueue()

print("STUDENT TEACHER PROBLEM")
print()
print("Select Mode")
print("0. Pre defined mode")
print("1. Automatic Mode")
print("2. Mannual Mode")
print("Any other key to exit ONLY NUMERICS")
while True:
    try:
        ip_var = int(input("--> "))
        break
    except ValueError:
        pass
#ip_var = 1
if ip_var == 0:
    print("Predefined Mode Selected")
    print()
    Tat1, Tbt1 = 1, 2
    Tat2, Tbt2 = 2, 2
    Tat3, Tbt3 = 3, 2
    Tat4, Tbt4 = 14, 3
    TQ.enqueue(1, Tat1, Tbt1)
    TQ.enqueue(2, Tat2, Tbt2)
    TQ.enqueue(3, Tat3, Tbt3)
    TQ.enqueue(4, Tat4, Tbt4)
    Sat1, Sbt1 = 1, 2
    Sat2, Sbt2 = 2, 2
    SQ.enqueue(1, Sat1, Sbt1)
    SQ.enqueue(2, Sat2, Sbt2)
    teachers = TQ.size()
    students = SQ.size()
elif ip_var == 1:
    print("Automatic Mode Selected")
    auto = 1
    if auto == 1:
        tchr = random.randint(1, 51)
        lston = 0
        for xx in range (tchr):
            arrival_time = random.randint(1, 51)
            if arrival_time < lston:
                while True:
                    arrival_time = random.randint(1, 51)
                    if arrival_time >= lston:
                        break
            lston = arrival_time
            burst_time = random.randint(1, 51)
            idno = xx+1
            TQ.enqueue(idno, arrival_time, burst_time)
            print("Teacher",idno, "  AT:",arrival_time, " BT:",burst_time)

        stdnt = random.randint(1, 51)
        lston = 0
        for xx in range (stdnt):
            arrival_time = random.randint(1, 51)
            if arrival_time < lston:
                while True:
                    arrival_time = random.randint(1, 51)
                    if arrival_time >= lston:
                        break
            lston = arrival_time
            burst_time = random.randint(1, 51)
            idno = xx+1
            SQ.enqueue(idno, arrival_time, burst_time)
            print("Student",idno, "  AT:",arrival_time, " BT:",burst_time)
        teachers = TQ.size()
        students = SQ.size()
        print("Teachers: ", teachers)
        print("Students: ", students)

elif ip_var == 2:
    print("User Mode Selected")
    print()
```

```
while True:
    try:
        teachers = int(input("Enter the number of Teachers in the queue: ", ))
        break
    except ValueError:
        pass
t_data = []
last_time = 0
if teachers >= 0:
    for i in range(teachers):
        print("Enter Arrival Time for Teacher ",i+1, end = "")
        while True:
            try:
                AT = int(input())
                break
            except ValueError:
                pass
        if (AT < last_time):
            while True:
                print("AT can't be less then previous arrival time")
                print("Enter Arrival Time for Teacher ",i+1, end = "")
                while True:
                    try:
                        AT = int(input())
                        break
                    except ValueError:
                        pass
                if last_time <= AT:
                    break
        last_time = AT
        print("Enter Burst Time for Teacher ",i+1, end = "")
        while True:
            try:
                BT = int(input())
                break
            except ValueError:
                pass
        if BT <= 0:
            while True:
                print("Error: BT can't be less than 1 ##Min BT req: 1")
                print("Enter Burst Time for Teacher ",i+1, end = "")
                while True:
                    try:
                        BT = int(input())
                        break
                    except ValueError:
                        pass
                if BT > 0:
                    break
        temp_list = []
        temp_list.append(AT)
        temp_list.append(BT)
        t_data.append(temp_list)
else:
    while True:
        print("Number of Teachers can't be less than 0")
        while True:
            try:
                teachers = int(input("Pleas re-enter the number of Teachers: "))
                break
            except ValueError:
                pass
        if teachers >= 0:
            for i in range(teachers):
```

```
print("Enter Arrival Time for Teacher ",i+1, end = "")
while True:
    try:
        AT = int(input())
        break
    except ValueError:
        pass
if (AT < last_time):
    while True:
        print("AT can't be less then previous arrival time")
        print("Enter Arrival Time for Teacher ",i+1, end = "")
        while True:
            try:
                AT = int(input())
                break
            except ValueError:
                pass
        if last_time <= AT:
            break
last_time = AT
print("Enter Burst Time for Teacher ",i+1, end = "")
while True:
    try:
        BT = int(input())
        break
    except ValueError:
        pass
if BT <= 0:
    while True:
        print("Error: BT can't be less than 1 ##Min BT req: 1")
        print("Enter Burst Time for Teacher ",i+1, end = "")
        while True:
```

```
                            try:
                                BT = int(input())
                                break
                            except ValueError:
                                pass
                        if BT > 0:
                            break
                temp_list = []
                temp_list.append(AT)
                temp_list.append(BT)
                t_data.append(temp_list)
            break

while True:
    try:
        students = int(input("Enter the nubers of Students in the queue: ", ))
        break
    except ValueError:
        pass
s_data = []
last_time = 0
if students >= 0:
    for i in range(students):
        print("Enter Arrival Time for Student ",i+1, end = "")
        while True:
            try:
                AT = int(input())
                break
            except ValueError:
                pass
        if (AT < last_time):
            while True:
```

```
        print("AT can't be less then previous arrival time")
        print("Enter Arrival Time for Student ",i+1, end = "")
        while True:
            try:
                AT = int(input())
                break
            except ValueError:
                pass
        if last_time <= AT:
            break
last_time = AT
print("Enter Burst Time for Student ",i+1, end = "")
while True:
    try:
        BT = int(input())
        break
    except ValueError:
        pass
if BT <= 0:
    while True:
        print("Error: BT can't be less than 1 ##Min BT req: 1")
        print("Enter Burst Time for Student ",i+1, end = "")
        while True:
            try:
                BT = int(input())
                break
            except ValueError:
                pass
        if BT > 0:
            break
temp_list = []
temp_list.append(AT)
```

```
        temp_list.append(BT)
        s_data.append(temp_list)
else:
    while True:
        print("Number of Students can't be less than 0")
        while True:
            try:
                students = int(input("Pleas re-enter the number of Students: "))
                break
            except ValueError:
                pass
        if students >= 0:
            for i in range(students):
                print("Enter Arrival Time for Student ",i+1, end = "")
                while True:
                    try:
                        AT = int(input())
                        break
                    except ValueError:
                        pass
                if (AT < last_time):
                    while True:
                        print("AT can't be less then previous arrival time")
                        print("Enter Arrival Time for Student ",i+1, end = "")
                        while True:
                            try:
                                AT = int(input())
                                break
                            except ValueError:
                                pass
                        if last_time <= AT:
                            break
                last_time = AT
                print("Enter Burst Time for Student ",i+1, end = "")
                while True:
                    try:
                        BT = int(input())
                        break
                    except ValueError:
                        pass
                if BT <= 0:
                    while True:
                        print("Error: BT can't be less than 1 ##Min BT req: 1")
                        print("Enter Burst Time for Student ",i+1, end = "")
                        while True:
                            try:
                                BT = int(input())
                                break
                            except ValueError:
                                pass
                        if BT > 0:
                            break
                temp_list = []
                temp_list.append(AT)
                temp_list.append(BT)
                s_data.append(temp_list)
            break

    for i in range (teachers):
        TQ.enqueue(i+1, t_data[i][0], t_data[i][1])
    for i in range(students):
        SQ.enqueue(i+1, s_data[i][0], s_data[i][1])
else:
    exit()
```

```
maxlen =teachers+students
student_priority = 0
if SQ.isEmpty() or TQ.isEmpty():
    if SQ.isEmpty() and TQ.isEmpty():
        print("Teacher and Student Queues are EMPTY")
    elif SQ.isEmpty():
        if TQ.isEmpty() != True:
            curr_time = TQ.head()
        else:
            print("Both the Queues are EMPTY")
    elif TQ.isEmpty():
        curr_time = SQ.head()
else:
    curr_time = min(SQ.head(), TQ.head())

t = teachers
s = students
j = 0
k = 0

'''
print("No of teachers: ", t)
print("No of students: ", s)
print("AT of first student is ", SQ.items[s-1][1])
print("AT of first teacher is ", TQ.items[t-1][1])

'''

for i in range(maxlen):
    if (SQ.isEmpty()):
        for i in range (teachers):
```

```
        if TQ.isEmpty() == False:
            print("Teacher ",TQ.id_no()," issued book")
            curr_time += TQ.burst_time()
            TQ.dequeue()
            break
elif TQ.isEmpty():
    for i in range (students):
        if SQ.isEmpty() == False:
            print("Student ",SQ.id_no()," issued book")
            curr_time += SQ.burst_time()
            SQ.dequeue()
            break
elif student_priority == 2:
        print("Student ",SQ.id_no()," issued book")
        curr_time += SQ.burst_time()
        student_priority = 0
        SQ.dequeue()
else:
    tchr = TQ.head()
    stdnt = SQ.head()
    if tchr <= stdnt:
        if curr_time >= stdnt:
            student_priority += 1
        print("Teacher ", TQ.id_no()," issued book. Student Priority: ", student_priority)
        curr_time += TQ.burst_time()
        TQ.dequeue()
    elif tchr > stdnt:
        if curr_time >= tchr:
            student_priority += 1
            curr_time += TQ.burst_time()
            print("Teacher ", TQ.id_no()," issued book. Student Priority: ", student_priority)
            TQ.dequeue()
```

```
else:
    curr_time += SQ.burst_time()
    print("Student ", SQ.id_no()," issued book")
    student_priority = 0
    SQ.dequeue()
```

Ques1. Explain the problem in terms of Operating System Concept?

Description:

There are two queues for two different type of processes which are represented by Teachers and Students and we may call the queues be TeacherQueue and StudentQueue which can enter in a library for issuing of books. But the issuer can handle only one request at a time either be it Student or Teacher. If a Student is already in the line and issuing a book than if a teacher comes than that Teacher will be the second person to get the book issued. But if a Teacher is already in the queue and a student and a teacher comes together in their queues. The teacher will be the one who will be given the priority to get the book issued. A student may wait if a Teacher is already in the queue. This situation may lead to aging of Student so the task was to minimize the waiting time of Student.

Ques2. Write the algorithm for proposed solution for the assigned problem.

Algorithm:

```
SET maxlen = len(Student Queue) + len(Teacher Queue)
for i in range(maxlen): #Iterate the loop in the range of maxlen
    if (SQ.isEmpty()): # Check if Student Queue is empty
        for i in range (teachers):
            if TQ.isEmpty() == False: #Check if Teacher Queue is not empty
                print("Teacher ",TQ.id_no()," issued book")
                curr_time += TQ.burst_time()
                TQ.dequeue()
                break
    elif TQ.isEmpty():
        for i in range (students):
            if SQ.isEmpty() == False:
                print("Student ",SQ.id_no()," issued book")
                curr_time += SQ.burst_time()
                SQ.dequeue()
                break
    elif student_priority == 2:
            print("Student ",SQ.id_no()," issued book")
            curr_time += SQ.burst_time()
            student_priority = 0
        SQ.dequeue()
  else:
     tchr = TQ.head()
     stdnt = SQ.head()
     if tchr <= stdnt:
        if curr_time >= stdnt:
           student_priority += 1
        print("Teacher ", TQ.id_no()," issued book. Student Priority: ", student_priority)
        curr_time += TQ.burst_time()
        TQ.dequeue()
     elif tchr > stdnt:
        if curr_time >= tchr:
           student_priority += 1
           curr_time += TQ.burst_time()
           print("Teacher ", TQ.id_no()," issued book. Student Priority: ", student_priority)
           TQ.dequeue()
        else:
           curr_time += SQ.burst_time()
           print("Student ", SQ.id_no()," issued book")
           student_priority = 0
           SQ.dequeue()
```

Ques3. Calculate complexity of implemented algorithm.

Complexity:

```
for i in range(maxlen): // O(N)
  if (SQ.isEmpty()): // O(1)
     for i in range (teachers): // O(N)
        if TQ.isEmpty() == False: // O(1)
           print("Teacher ",TQ.id_no()," issued book") // O(1)
           curr_time += TQ.burst_time() // O(1)
           TQ.dequeue()// O(1)
```

```
            break
elif TQ.isEmpty():// O(1)
    for i in range (students): // O(N)
        if SQ.isEmpty() == False: // O(1)
            print("Student ",SQ.id_no()," issued book") // O(1)
            curr_time += SQ.burst_time() // O(1)
            SQ.dequeue() // O(1)
            break // O(1)
elif student_priority == 2: // O(1)
        print("Student ",SQ.id_no()," issued book") // O(1)
        curr_time += SQ.burst_time()// O(1)
        student_priority = 0 // O(1)
        SQ.dequeue() // O(1)
else: // O(1)
    tchr = TQ.head() // O(1)
    stdnt = SQ.head() // O(1)
    if tchr <= stdnt: // O(1)
        if curr_time >= stdnt: // O(1)
            student_priority += 1 // O(1)
        print("Teacher ", TQ.id_no(),"issued book. Student Priority:",student_priority) // O(1)
        curr_time += TQ.burst_time()// O(1)
        TQ.dequeue()// O(1)
    elif tchr > stdnt: // O(1)
        if curr_time >= tchr: // O(1)
            student_priority += 1 // O(1)
            curr_time += TQ.burst_time()// O(1)
            print("Teacher",TQ.id_no(),"issued book.Student Priority:",student_priority)// O(1)
            TQ.dequeue()// O(1)
        else: // O(1)
            curr_time += SQ.burst_time() // O(1)
            print("Student ", SQ.id_no()," issued book") // O(1)
            student_priority = 0 // O(1)
```

SQ.dequeue() // O(1)

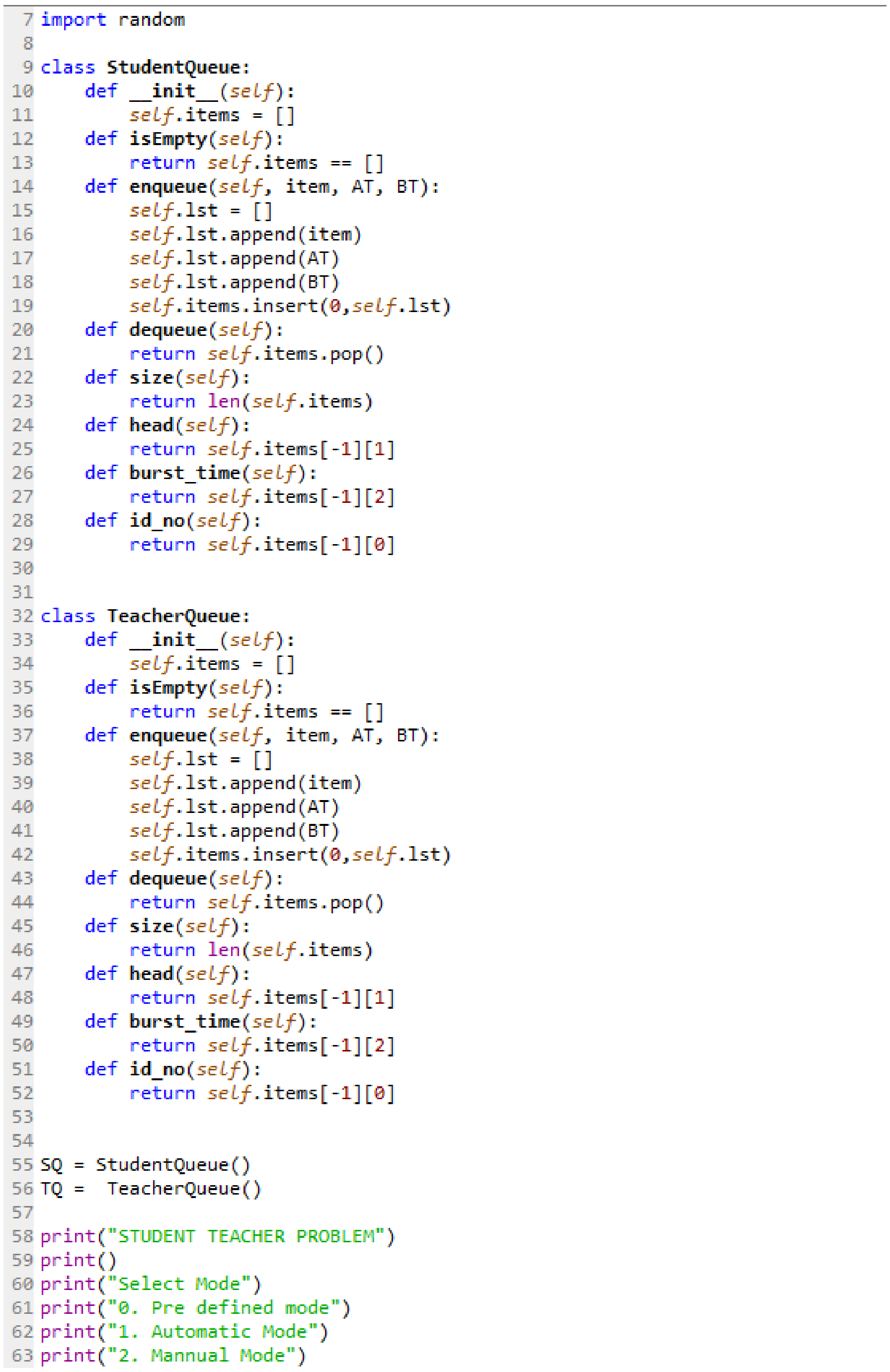
Total Complexity: O (len|Student Queue| + len|Teacher Queue|) → O(maxlen) → O(N)

Ques 4. Explain all the constraints given in the problem. Attach the code snippet of the implemented constraint.

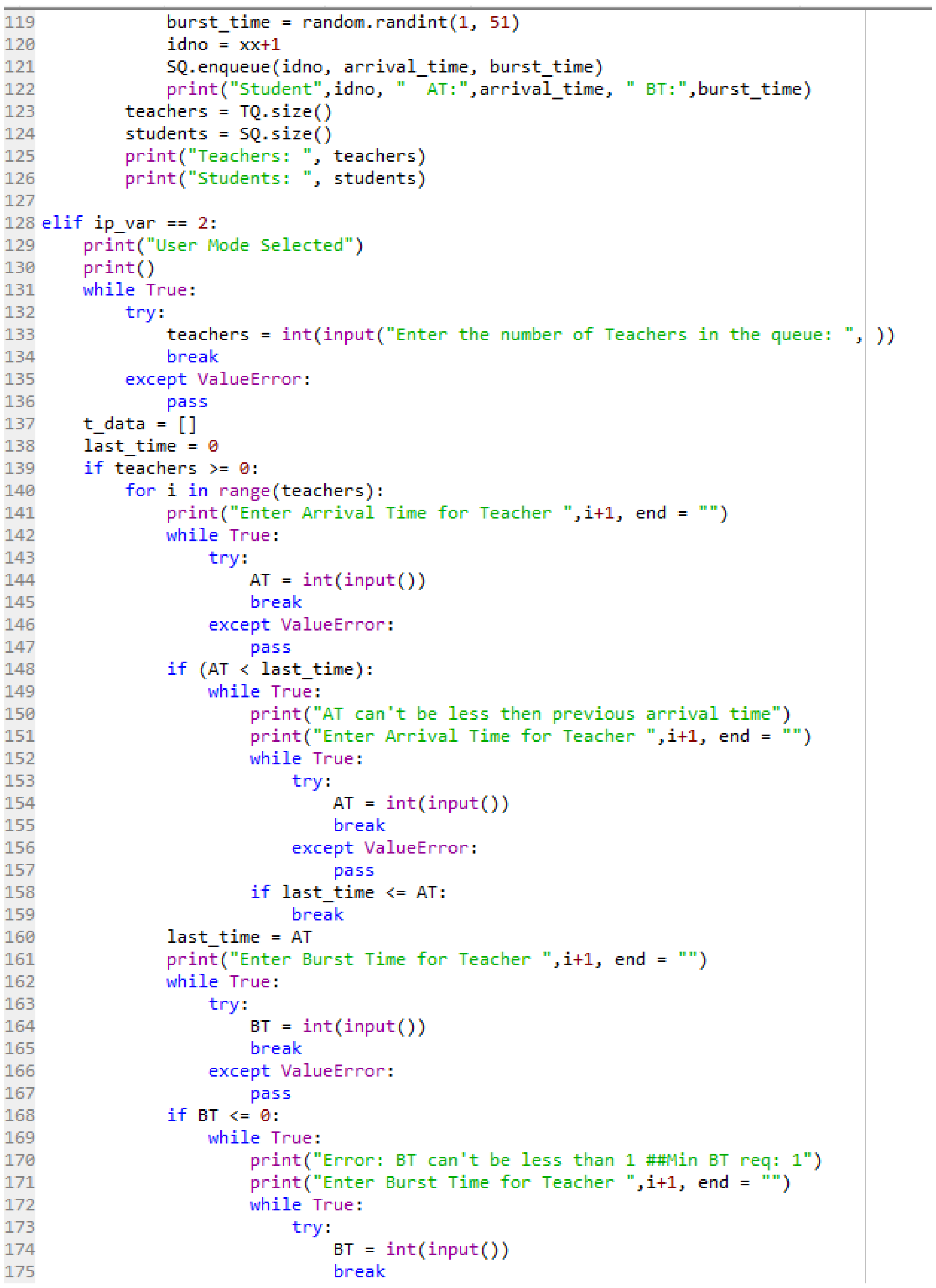
Code Snippet:

For Adding items into a python list on a regular 32bit system, this is 536,870,912 elements.

i.e. for appending items into the list, the maximum no of adding Teacher/Student in the queue is 536,870,912.



```python
import random

class StudentQueue:
    def __init__(self):
        self.items = []
    def isEmpty(self):
        return self.items == []
    def enqueue(self, item, AT, BT):
        self.lst = []
        self.lst.append(item)
        self.lst.append(AT)
        self.lst.append(BT)
        self.items.insert(0,self.lst)
    def dequeue(self):
        return self.items.pop()
    def size(self):
        return len(self.items)
    def head(self):
        return self.items[-1][1]
    def burst_time(self):
        return self.items[-1][2]
    def id_no(self):
        return self.items[-1][0]


class TeacherQueue:
    def __init__(self):
        self.items = []
    def isEmpty(self):
        return self.items == []
    def enqueue(self, item, AT, BT):
        self.lst = []
        self.lst.append(item)
        self.lst.append(AT)
        self.lst.append(BT)
        self.items.insert(0,self.lst)
    def dequeue(self):
        return self.items.pop()
    def size(self):
        return len(self.items)
    def head(self):
        return self.items[-1][1]
    def burst_time(self):
        return self.items[-1][2]
    def id_no(self):
        return self.items[-1][0]


SQ = StudentQueue()
TQ =  TeacherQueue()

print("STUDENT TEACHER PROBLEM")
print()
print("Select Mode")
print("0. Pre defined mode")
print("1. Automatic Mode")
print("2. Mannual Mode")
print("2. Mannual Mode")
print("Any other key to exit ONLY NUMERICS")
while True:
    try:
        ip_var = int(input("--> "))
        break
    except ValueError:
        pass
#ip_var = 1
if ip_var == 0:
    print("Predefined Mode Selected")
    print()
    Tat1, Tbt1 = 1, 2
    Tat2, Tbt2 = 2, 2
    Tat3, Tbt3 = 3, 2
    Tat4, Tbt4 = 14, 3
    TQ.enqueue(1, Tat1, Tbt1)
    TQ.enqueue(2, Tat2, Tbt2)
    TQ.enqueue(3, Tat3, Tbt3)
    TQ.enqueue(4, Tat4, Tbt4)
    Sat1, Sbt1 = 1, 2
    Sat2, Sbt2 = 2, 2
    SQ.enqueue(1, Sat1, Sbt1)
    SQ.enqueue(2, Sat2, Sbt2)
    teachers = TQ.size()
    students = SQ.size()
elif ip_var == 1:
    print("Automatic Mode Selected")
    auto = 1
    if auto == 1:
        print("Small Mode Selected")
        tchr = random.randint(1, 51)
        lston = 0
        for xx in range (tchr):
            arrival_time = random.randint(1, 51)
            if arrival_time < lston:
                while True:
                    arrival_time = random.randint(1, 51)
                    if arrival_time >= lston:
                        break
            lston = arrival_time
            burst_time = random.randint(1, 51)
            idno = xx+1
            TQ.enqueue(idno, arrival_time, burst_time)
            print("Teacher",idno, "  AT:",arrival_time, " BT:",burst_time)

        stdnt = random.randint(1, 51)
        lston = 0
        for xx in range (stdnt):
            arrival_time = random.randint(1, 51)
            if arrival_time < lston:
                while True:
                    arrival_time = random.randint(1, 51)
                    if arrival_time >= lston:
                        break
            lston = arrival_time
            burst_time = random.randint(1, 51)
```



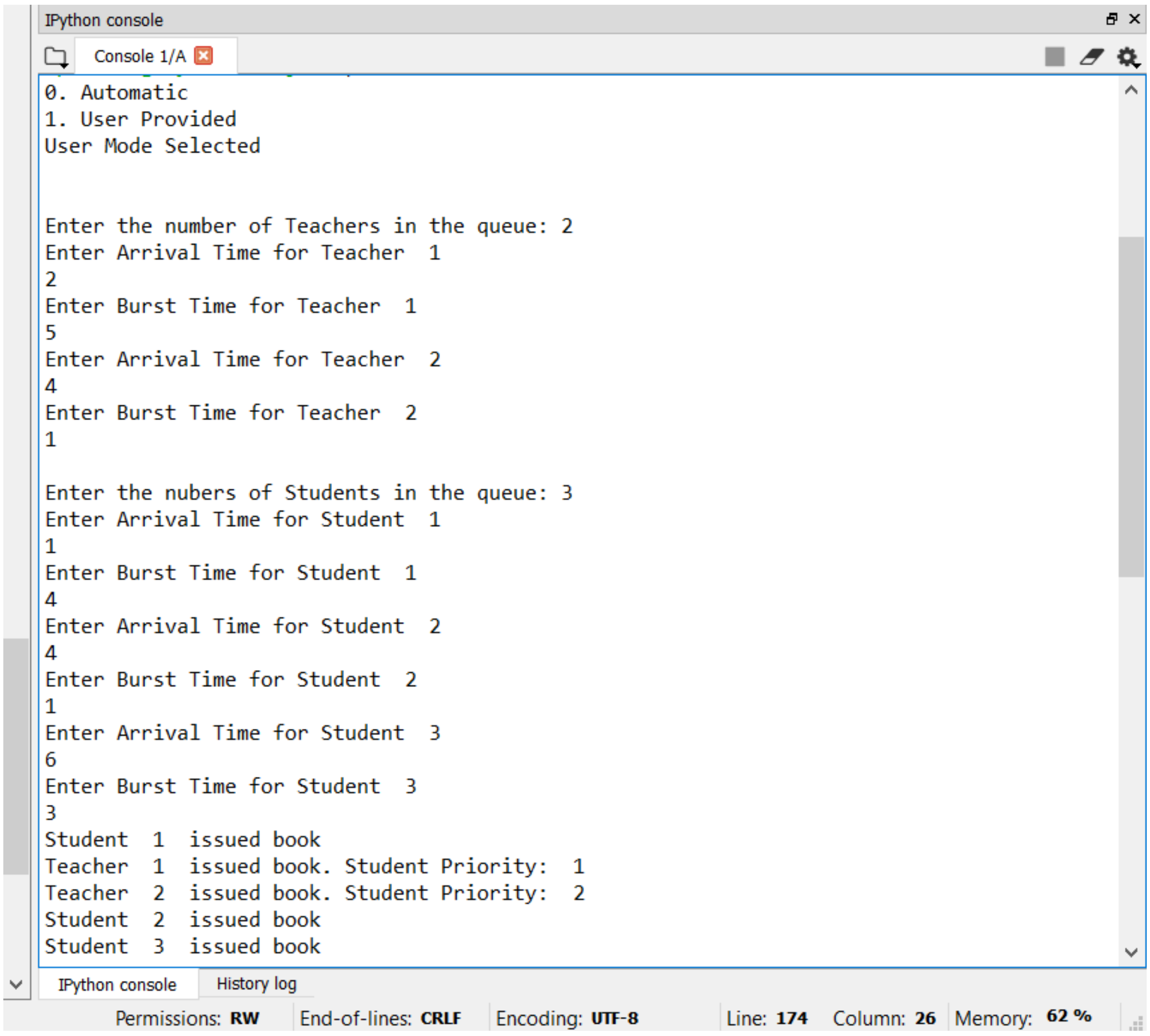
```
            burst_time = random.randint(1, 51)
            idno = xx+1
            SQ.enqueue(idno, arrival_time, burst_time)
            print("Student",idno, "  AT:",arrival_time, " BT:",burst_time)
        teachers = TQ.size()
        students = SQ.size()
        print("Teachers: ", teachers)
        print("Students: ", students)

elif ip_var == 2:
    print("User Mode Selected")
    print()
    while True:
        try:
            teachers = int(input("Enter the number of Teachers in the queue: ", ))
            break
        except ValueError:
            pass
    t_data = []
    last_time = 0
    if teachers >= 0:
        for i in range(teachers):
            print("Enter Arrival Time for Teacher ",i+1, end = "")
            while True:
                try:
                    AT = int(input())
                    break
                except ValueError:
                    pass
            if (AT < last_time):
                while True:
                    print("AT can't be less then previous arrival time")
                    print("Enter Arrival Time for Teacher ",i+1, end = "")
                    while True:
                        try:
                            AT = int(input())
                            break
                        except ValueError:
                            pass
                    if last_time <= AT:
                        break
            last_time = AT
            print("Enter Burst Time for Teacher ",i+1, end = "")
            while True:
                try:
                    BT = int(input())
                    break
                except ValueError:
                    pass
            if BT <= 0:
                while True:
                    print("Error: BT can't be less than 1 ##Min BT req: 1")
                    print("Enter Burst Time for Teacher ",i+1, end = "")
                    while True:
                        try:
                            BT = int(input())
                            break
```

```python
                            BT = int(input())
                            break
                        except ValueError:
                            pass
                    if BT > 0:
                        break
            temp_list = []
            temp_list.append(AT)
            temp_list.append(BT)
            t_data.append(temp_list)
    else:
        while True:
            print("Number of Teachers can't be less than 0")
            while True:
                try:
                    teachers = int(input("Pleas re-enter the number of Teachers: "))
                    break
                except ValueError:
                    pass
            if teachers >= 0:
                for i in range(teachers):
                    print("Enter Arrival Time for Teacher ",i+1, end = "")
                    while True:
                        try:
                            AT = int(input())
                            break
                        except ValueError:
                            pass
                    if (AT < last_time):
                        while True:
                            print("AT can't be less then previous arrival time")
                            print("Enter Arrival Time for Teacher ",i+1, end = "")
                            while True:
                                try:
                                    AT = int(input())
                                    break
                                except ValueError:
                                    pass
                            if last_time <= AT:
                                break
                    last_time = AT
                    print("Enter Burst Time for Teacher ",i+1, end = "")
                    while True:
                        try:
                            BT = int(input())
                            break
                        except ValueError:
                            pass
                    if BT <= 0:
                        while True:
                            print("Error: BT can't be less than 1 ##Min BT req: 1")
                            print("Enter Burst Time for Teacher ",i+1, end = "")
                            while True:
                                try:
                                    BT = int(input())
                                    break
                                except ValueError:
```

```
                                pass
                        if BT > 0:
                            break
                temp_list = []
                temp_list.append(AT)
                temp_list.append(BT)
                t_data.append(temp_list)
            break

    while True:
        try:
            students = int(input("Enter the nubers of Students in the queue: ", ))
            break
        except ValueError:
            pass
    s_data = []
    last_time = 0
    if students >= 0:
        for i in range(students):
            print("Enter Arrival Time for Student ",i+1, end = "")
            while True:
                try:
                    AT = int(input())
                    break
                except ValueError:
                    pass
            if (AT < last_time):
                while True:
                    print("AT can't be less then previous arrival time")
                    print("Enter Arrival Time for Student ",i+1, end = "")
                    while True:
                        try:
                            AT = int(input())
                            break
                        except ValueError:
                            pass
                    if last_time <= AT:
                        break
            last_time = AT
            print("Enter Burst Time for Student ",i+1, end = "")
            while True:
                try:
                    BT = int(input())
                    break
                except ValueError:
                    pass
            if BT <= 0:
                while True:
                    print("Error: BT can't be less than 1 ##Min BT req: 1")
                    print("Enter Burst Time for Student ",i+1, end = "")
                    while True:
                        try:
                            BT = int(input())
                            break
                        except ValueError:
                            pass
                    if BT > 0:
                        break
            temp_list = []
            temp_list.append(AT)
            temp_list.append(BT)
            s_data.append(temp_list)
    else:
        while True:
            print("Number of Students can't be less than 0")
            while True:
                try:
                    students = int(input("Pleas re-enter the number of Students: "))
                    break
                except ValueError:
                    pass
            if students >= 0:
                for i in range(students):
                    print("Enter Arrival Time for Student ",i+1, end = "")
                    while True:
                        try:
                            AT = int(input())
                            break
                        except ValueError:
                            pass
                    if (AT < last_time):
                        while True:
                            print("AT can't be less then previous arrival time")
                            print("Enter Arrival Time for Student ",i+1, end = "")
                            while True:
                                try:
                                    AT = int(input())
                                    break
                                except ValueError:
                                    pass
                            if last_time <= AT:
                                break
                    last_time = AT
                    print("Enter Burst Time for Student ",i+1, end = "")
                    while True:
                        try:
                            BT = int(input())
                            break
                        except ValueError:
                            pass
                    if BT <= 0:
                        while True:
                            print("Error: BT can't be less than 1 ##Min BT req: 1")
                            print("Enter Burst Time for Student ",i+1, end = "")
                            while True:
                                try:
                                    BT = int(input())
                                    break
                                except ValueError:
                                    pass
                            if BT > 0:
                                break
                    temp_list = []
                    temp_list.append(AT)
```

```python
                    temp_list.append(AT)
                    temp_list.append(BT)
                    s_data.append(temp_list)
                break

    for i in range (teachers):
        TQ.enqueue(i+1, t_data[i][0], t_data[i][1])
    for i in range(students):
        SQ.enqueue(i+1, s_data[i][0], s_data[i][1])
else:
    exit()

maxlen =teachers+students
student_priority = 0
if SQ.isEmpty() or TQ.isEmpty():
    if SQ.isEmpty() and TQ.isEmpty():
        print("Teacher and Student Queues are EMPTY")
    elif SQ.isEmpty():
        if TQ.isEmpty() != True:
            curr_time = TQ.head()
        else:
            print("Both the Queues are EMPTY")
    elif TQ.isEmpty():
        curr_time = SQ.head()
else:
    curr_time = min(SQ.head(), TQ.head())

t = teachers
s = students
j = 0
k = 0

'''
print("No of teachers: ", t)
print("No of students: ", s)
print("AT of first student is ", SQ.items[s-1][1])
print("AT of first teacher is ", TQ.items[t-1][1])
'''

for i in range(maxlen):
    if (SQ.isEmpty()):
        for i in range (teachers):
            if TQ.isEmpty() == False:
                print("Teacher ",TQ.id_no()," issued book")
                curr_time += TQ.burst_time()
                TQ.dequeue()
                break
    elif TQ.isEmpty():
        for i in range (students):
            if SQ.isEmpty() == False:
                print("Student ",SQ.id_no()," issued book")
                curr_time += SQ.burst_time()
                SQ.dequeue()
                break
    elif student_priority == 2:
            print("Student ",SQ.id_no()," issued book")
```

```

t = teachers
s = students
j = 0
k = 0

'''
print("No of teachers: ", t)
print("No of students: ", s)
print("AT of first student is ", SQ.items[s-1][1])
print("AT of first teacher is ", TQ.items[t-1][1])

'''

for i in range(maxlen):
    if (SQ.isEmpty()):
        for i in range (teachers):
            if TQ.isEmpty() == False:
                print("Teacher ",TQ.id_no()," issued book")
                curr_time += TQ.burst_time()
                TQ.dequeue()
                break
    elif TQ.isEmpty():
        for i in range (students):
            if SQ.isEmpty() == False:
                print("Student ",SQ.id_no()," issued book")
                curr_time += SQ.burst_time()
                SQ.dequeue()
                break
    elif student_priority == 2:
            print("Student ",SQ.id_no()," issued book")
            curr_time += SQ.burst_time()
            student_priority = 0
            SQ.dequeue()
    else:
        tchr = TQ.head()
        stdnt = SQ.head()
        if tchr <= stdnt:
            if curr_time >= stdnt:
                student_priority += 1
            print("Teacher ", TQ.id_no()," issued book. Student Priority: ", student_priority)
            curr_time += TQ.burst_time()
            TQ.dequeue()
        elif tchr > stdnt:
            if curr_time >= tchr:
                student_priority += 1
                curr_time += TQ.burst_time()
                print("Teacher ", TQ.id_no()," issued book. Student Priority: ", student_priority)
                TQ.dequeue()
            else:
                curr_time += SQ.burst_time()
                print("Student ", SQ.id_no()," issued book")
                student_priority = 0
                SQ.dequeue()


```



```
0. Automatic
1. User Provided
User Mode Selected


Enter the number of Teachers in the queue: 2
Enter Arrival Time for Teacher  1
2
Enter Burst Time for Teacher  1
5
Enter Arrival Time for Teacher  2
4
Enter Burst Time for Teacher  2
1

Enter the nubers of Students in the queue: 3
Enter Arrival Time for Student  1
1
Enter Burst Time for Student  1
4
Enter Arrival Time for Student  2
4
Enter Burst Time for Student  2
1
Enter Arrival Time for Student  3
6
Enter Burst Time for Student  3
3
Student  1  issued book
Teacher  1  issued book. Student Priority:  1
Teacher  2  issued book. Student Priority:  2
Student  2  issued book
Student  3  issued book
```

Permissions: **RW** End-of-lines: **CRLF** Encoding: **UTF-8** Line: **174** Column: **26** Memory: **62 %**

Ques 5. If you implemented any additional algorithm to support the solution, explain the need and usage of same.

Description:

Algorithm used in this code just looks for the first person to arrive at the counter and 4 possibilities can occur –

1. Both Teacher and Student arrives at the same time. Than priority will be gives to the Teacher and he will be the one to get the book issued. In case if another Teacher just shows up than the priority will be given to Teacher and the Student at the front of the Queue have to wait to get served. And Further if one more Teacher shows up than the Student waiting will be given the priority to get served.
   *A student can only wait for 2 or 3 Teachers, but after that the student will run
2. If no Teacher and Student are in the queue than no person will be served.
3. If 0 Students and N Teachers. Than those N teachers will be served one by one and vice-versa.

4. The person who arrives first will be served first.

Ques 6. Explain the boundary conditions of the implemented code.

Description:

If a Student and a teacher arrives at same after a Teacher than the student can only wait for one more Teacher only, and after that Student will issue the book.

The Arrival Time for a Student/Teacher can't be less than the previous Arrival Time.

The Burst Time for a Student/ Teacher can't be less than 1.

A user should not press any character rather than numeric otherwise the user will be re-prompted for the input.

Ques 7. Explain all the test cases applied on the solution of assigned problem.

Description:

S.No	**Condition**	**Expected Result**	**Actual Result**
1.	User Selects Mode 1.	Predefined Mode opens.	Predefined Mode opens.
2.	User Selects Mode 2.	Automatic Mode opens.	Automatic Mode opens.
3.	User Selects Mode 3.	Manual Mode opens.	Manual Mode opens.
4.	User Hits Enter or any other key rather than NUMERICS.	It Re-prompts the user.	It Re-prompts the user.
5.	User Select any NUMERIC rather that 0, 1 and 2.	Console Terminates.	Console Terminates.
6.	If 0 Teacher and N students.	N Students issues book one by one.	N Students issues book one by one.
7.	If N Teachers and 0 students.	N Teachers issues book one by one.	N Teachers issues book one by one.
8.	If N Teachers arrive at the same time and 0 students.	The teacher which is in front of the queue will be served first.	The teacher which is in front of the queue will be served first.
9.	If 0 Teachers arrive at the same time and N students.	The student which is in front of the queue will be served first	The student which is in front of the queue will be served first.

10.	If Arrival Time of any Student/Teacher is less than the previous Student/Teacher Arrival Time.	It should re prompt the user.	It should re prompt the user.
11.	If 3 Teachers arrive at the counter than 1 student arrives simultaneously with 3rd Teacher at the counter and after that 2 more teacher arrives.	The student is served after the 4th Teacher.	The student is served after 4th Teacher.
12.	If 3 Student arrive first and a Teacher arrives at the same time with 2nd Student at the counter and after that 4 more Student arrives.	The teacher is served after the 1st student and the issuing goes on until the Student Queue is empty.	The teacher is served after the 1st student and the issuing goes on until the Student Queue is empty.

13.	Total Teacher are 3
	Total Student are 2
	Taking BT[T/S] = 2
	
	AT[T1] = 0 | AT[S1] = 0
	AT[T2] = 1 | AT[S2] = 1
	AT[T3] = 2
		Teacher 1 issued book.
		Teacher 2 issued book.
		Student 1 issued book
		Teacher 3 issued book.
		Student 2 issued book
			Teacher 1 issued book.
			Teacher 2 issued book.
			Student 1 issued book
			Teacher 3 issued book.
			Student 2 issued book

14.	Total Teacher are 4 Total Student are 4 Tno | AT | BT | Sno | AT | BT 1. | 1 | 2 | 1. | 0 | 2 2. | 3 | 1 | 2. | 1 | 1 3. | 4 | 2 | 3. | 4 | 2 4. | 10 | 3 | 4. | 7 | 1	Student 1 issued book Teacher 1 issued book. Teacher 2 issued book. Student 2 issued book Teacher 3 issued book. Student 3 issued book Teacher 4 issued book. Student 4 issued book	Student 1 issued book Teacher 1 issued book. Teacher 2 issued book. Student 2 issued book Teacher 3 issued book. Student 3 issued book Teacher 4 issued book. Student 4 issued book

Ques 8. Have you made minimum 5 revisions of solution on GitHub?

GitHub Link: https://github.com/akhildhiman7/Student-Teacher-Problem.git