



LOVELY
PROFESSIONAL
UNIVERSITY

Transforming Education Transforming India

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

CODE: (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: $O(\text{len}|\text{Student Queue}| + \text{len}|\text{Teacher Queue}|) \rightarrow O(\text{maxlen}) \rightarrow O(N)$

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.

```

7 import random
8
9 class StudentQueue:
10     def __init__(self):
11         self.items = []
12     def isEmpty(self):
13         return self.items == []
14     def enqueue(self, item, AT, BT):
15         self.lst = []
16         self.lst.append(item)
17         self.lst.append(AT)
18         self.lst.append(BT)
19         self.items.insert(0, self.lst)
20     def dequeue(self):
21         return self.items.pop()
22     def size(self):
23         return len(self.items)
24     def head(self):
25         return self.items[-1][1]
26     def burst_time(self):
27         return self.items[-1][2]
28     def id_no(self):
29         return self.items[-1][0]
30
31
32 class TeacherQueue:
33     def __init__(self):
34         self.items = []
35     def isEmpty(self):
36         return self.items == []
37     def enqueue(self, item, AT, BT):
38         self.lst = []
39         self.lst.append(item)
40         self.lst.append(AT)
41         self.lst.append(BT)
42         self.items.insert(0, self.lst)
43     def dequeue(self):
44         return self.items.pop()
45     def size(self):
46         return len(self.items)
47     def head(self):
48         return self.items[-1][1]
49     def burst_time(self):
50         return self.items[-1][2]
51     def id_no(self):
52         return self.items[-1][0]
53
54
55 SQ = StudentQueue()
56 TQ = TeacherQueue()
57
58 print("STUDENT TEACHER PROBLEM")
59 print()
60 print("Select Mode")
61 print("0. Pre defined mode")
62 print("1. Automatic Mode")
63 print("2. Mannual Mode")

```

```

63 print("2. Mannual Mode")
64 print("Any other key to exit ONLY NUMERICS")
65 while True:
66     try:
67         ip_var = int(input("--> "))
68         break
69     except ValueError:
70         pass
71 #ip_var = 1
72 if ip_var == 0:
73     print("Predefined Mode Selected")
74     print()
75     Tat1, Tbt1 = 1, 2
76     Tat2, Tbt2 = 2, 2
77     Tat3, Tbt3 = 3, 2
78     Tat4, Tbt4 = 14, 3
79     TQ.enqueue(1, Tat1, Tbt1)
80     TQ.enqueue(2, Tat2, Tbt2)
81     TQ.enqueue(3, Tat3, Tbt3)
82     TQ.enqueue(4, Tat4, Tbt4)
83     Sat1, Sbt1 = 1, 2
84     Sat2, Sbt2 = 2, 2
85     SQ.enqueue(1, Sat1, Sbt1)
86     SQ.enqueue(2, Sat2, Sbt2)
87     teachers = TQ.size()
88     students = SQ.size()
89 elif ip_var == 1:
90     print("Automatic Mode Selected")
91     auto = 1
92     if auto == 1:
93         print("Small Mode Selected")
94         tchr = random.randint(1, 51)
95         lston = 0
96         for xx in range (tchr):
97             arrival_time = random.randint(1, 51)
98             if arrival_time < lston:
99                 while True:
100                     arrival_time = random.randint(1, 51)
101                     if arrival_time >= lston:
102                         break
103                 lston = arrival_time
104                 burst_time = random.randint(1, 51)
105                 idno = xx+1
106                 TQ.enqueue(idno, arrival_time, burst_time)
107                 print("Teacher",idno, " AT:",arrival_time, " BT:",burst_time)
108
109         stdnt = random.randint(1, 51)
110         lston = 0
111         for xx in range (stdnt):
112             arrival_time = random.randint(1, 51)
113             if arrival_time < lston:
114                 while True:
115                     arrival_time = random.randint(1, 51)
116                     if arrival_time >= lston:
117                         break
118                 lston = arrival_time
119                 burst_time = random.randint(1, 51)

```

```

119         burst_time = random.randint(1, 51)
120         idno = xx+1
121         SQ.enqueue(idno, arrival_time, burst_time)
122         print("Student",idno, " AT:",arrival_time, " BT:",burst_time)
123     teachers = TQ.size()
124     students = SQ.size()
125     print("Teachers: ", teachers)
126     print("Students: ", students)
127
128 elif ip_var == 2:
129     print("User Mode Selected")
130     print()
131     while True:
132         try:
133             teachers = int(input("Enter the number of Teachers in the queue: ", ))
134             break
135         except ValueError:
136             pass
137     t_data = []
138     last_time = 0
139     if teachers >= 0:
140         for i in range(teachers):
141             print("Enter Arrival Time for Teacher ",i+1, end = "")
142             while True:
143                 try:
144                     AT = int(input())
145                     break
146                 except ValueError:
147                     pass
148             if (AT < last_time):
149                 while True:
150                     print("AT can't be less then previous arrival time")
151                     print("Enter Arrival Time for Teacher ",i+1, end = "")
152                     while True:
153                         try:
154                             AT = int(input())
155                             break
156                         except ValueError:
157                             pass
158                     if last_time <= AT:
159                         break
160             last_time = AT
161             print("Enter Burst Time for Teacher ",i+1, end = "")
162             while True:
163                 try:
164                     BT = int(input())
165                     break
166                 except ValueError:
167                     pass
168             if BT <= 0:
169                 while True:
170                     print("Error: BT can't be less than 1 ##Min BT req: 1")
171                     print("Enter Burst Time for Teacher ",i+1, end = "")
172                     while True:
173                         try:
174                             BT = int(input())
175                             break

```

```

174         BT = int(input())
175         break
176     except ValueError:
177         pass
178     if BT > 0:
179         break
180     temp_list = []
181     temp_list.append(AT)
182     temp_list.append(BT)
183     t_data.append(temp_list)
184 else:
185     while True:
186         print("Number of Teachers can't be less than 0")
187         while True:
188             try:
189                 teachers = int(input("Pleas re-enter the number of Teachers: "))
190                 break
191             except ValueError:
192                 pass
193         if teachers >= 0:
194             for i in range(teachers):
195                 print("Enter Arrival Time for Teacher ",i+1, end = "")
196                 while True:
197                     try:
198                         AT = int(input())
199                         break
200                     except ValueError:
201                         pass
202                 if (AT < last_time):
203                     while True:
204                         print("AT can't be less then previous arrival time")
205                         print("Enter Arrival Time for Teacher ",i+1, end = "")
206                         while True:
207                             try:
208                                 AT = int(input())
209                                 break
210                             except ValueError:
211                                 pass
212                         if last_time <= AT:
213                             break
214                 last_time = AT
215                 print("Enter Burst Time for Teacher ",i+1, end = "")
216                 while True:
217                     try:
218                         BT = int(input())
219                         break
220                     except ValueError:
221                         pass
222                 if BT <= 0:
223                     while True:
224                         print("Error: BT can't be less than 1 ##Min BT req: 1")
225                         print("Enter Burst Time for Teacher ",i+1, end = "")
226                         while True:
227                             try:
228                                 BT = int(input())
229                                 break
230                             except ValueError:

```



```

231         pass
232         if BT > 0:
233             break
234         temp_list = []
235         temp_list.append(AT)
236         temp_list.append(BT)
237         t_data.append(temp_list)
238     break
239
240 while True:
241     try:
242         students = int(input("Enter the nubers of Students in the queue: ", ))
243         break
244     except ValueError:
245         pass
246 s_data = []
247 last_time = 0
248 if students >= 0:
249     for i in range(students):
250         print("Enter Arrival Time for Student ",i+1, end = "")
251         while True:
252             try:
253                 AT = int(input())
254                 break
255             except ValueError:
256                 pass
257         if (AT < last_time):
258             while True:
259                 print("AT can't be less then previous arrival time")
260                 print("Enter Arrival Time for Student ",i+1, end = "")
261                 while True:
262                     try:
263                         AT = int(input())
264                         break
265                     except ValueError:
266                         pass
267                 if last_time <= AT:
268                     break
269             last_time = AT
270         print("Enter Burst Time for Student ",i+1, end = "")
271         while True:
272             try:
273                 BT = int(input())
274                 break
275             except ValueError:
276                 pass
277         if BT <= 0:
278             while True:
279                 print("Error: BT can't be less than 1 ##Min BT req: 1")
280                 print("Enter Burst Time for Student ",i+1, end = "")
281                 while True:
282                     try:
283                         BT = int(input())
284                         break
285                     except ValueError:
286                         pass
287                 if BT > 0:

```

```

288         break
289     temp_list = []
290     temp_list.append(AT)
291     temp_list.append(BT)
292     s_data.append(temp_list)
293 else:
294     while True:
295         print("Number of Students can't be less than 0")
296         while True:
297             try:
298                 students = int(input("Pleas re-enter the number of Students: "))
299                 break
300             except ValueError:
301                 pass
302         if students >= 0:
303             for i in range(students):
304                 print("Enter Arrival Time for Student ",i+1, end = "")
305                 while True:
306                     try:
307                         AT = int(input())
308                         break
309                     except ValueError:
310                         pass
311                 if (AT < last_time):
312                     while True:
313                         print("AT can't be less then previous arrival time")
314                         print("Enter Arrival Time for Student ",i+1, end = "")
315                         while True:
316                             try:
317                                 AT = int(input())
318                                 break
319                             except ValueError:
320                                 pass
321                         if last_time <= AT:
322                             break
323                 last_time = AT
324                 print("Enter Burst Time for Student ",i+1, end = "")
325                 while True:
326                     try:
327                         BT = int(input())
328                         break
329                     except ValueError:
330                         pass
331                 if BT <= 0:
332                     while True:
333                         print("Error: BT can't be less than 1 ##Min BT req: 1")
334                         print("Enter Burst Time for Student ",i+1, end = "")
335                         while True:
336                             try:
337                                 BT = int(input())
338                                 break
339                             except ValueError:
340                                 pass
341                         if BT > 0:
342                             break
343             temp_list = []
344             temp_list.append(AT)

```

```

344         temp_list.append(AT)
345         temp_list.append(BT)
346         s_data.append(temp_list)
347         break
348
349     for i in range (teachers):
350         TQ.enqueue(i+1, t_data[i][0], t_data[i][1])
351     for i in range(students):
352         SQ.enqueue(i+1, s_data[i][0], s_data[i][1])
353 else:
354     exit()
355
356 maxlen =teachers+students
357 student_priority = 0
358 if SQ.isEmpty() or TQ.isEmpty():
359     if SQ.isEmpty() and TQ.isEmpty():
360         print("Teacher and Student Queues are EMPTY")
361     elif SQ.isEmpty():
362         if TQ.isEmpty() != True:
363             curr_time = TQ.head()
364         else:
365             print("Both the Queues are EMPTY")
366     elif TQ.isEmpty():
367         curr_time = SQ.head()
368 else:
369     curr_time = min(SQ.head(), TQ.head())
370
371 t = teachers
372 s = students
373 j = 0
374 k = 0
375 ...
376 ...
377 print("No of teachers: ", t)
378 print("No of students: ", s)
379 print("AT of first student is ", SQ.items[s-1][1])
380 print("AT of first teacher is ", TQ.items[t-1][1])
381 ...
382 ...
383
384 for i in range(maxlen):
385     if (SQ.isEmpty()):
386         for i in range (teachers):
387             if TQ.isEmpty() == False:
388                 print("Teacher ",TQ.id_no()," issued book")
389                 curr_time += TQ.burst_time()
390                 TQ.dequeue()
391                 break
392             elif TQ.isEmpty():
393                 for i in range (students):
394                     if SQ.isEmpty() == False:
395                         print("Student ",SQ.id_no()," issued book")
396                         curr_time += SQ.burst_time()
397                         SQ.dequeue()
398                         break
399             elif student_priority == 2:
400                 print("Student ",SQ.id_no()," issued book")

```

```

370
371 t = teachers
372 s = students
373 j = 0
374 k = 0
375
376 '''
377 print("No of teachers: ", t)
378 print("No of students: ", s)
379 print("AT of first student is ", SQ.items[s-1][1])
380 print("AT of first teacher is ", TQ.items[t-1][1])
381
382 '''
383
384 for i in range(maxlen):
385     if (SQ.isEmpty()):
386         for i in range (teachers):
387             if TQ.isEmpty() == False:
388                 print("Teacher ",TQ.id_no()," issued book")
389                 curr_time += TQ.burst_time()
390                 TQ.dequeue()
391                 break
392     elif TQ.isEmpty():
393         for i in range (students):
394             if SQ.isEmpty() == False:
395                 print("Student ",SQ.id_no()," issued book")
396                 curr_time += SQ.burst_time()
397                 SQ.dequeue()
398                 break
399     elif student_priority == 2:
400         print("Student ",SQ.id_no()," issued book")
401         curr_time += SQ.burst_time()
402         student_priority = 0
403         SQ.dequeue()
404     else:
405         tchr = TQ.head()
406         stdnt = SQ.head()
407         if tchr <= stdnt:
408             if curr_time >= stdnt:
409                 student_priority += 1
410                 print("Teacher ", TQ.id_no()," issued book. Student Priority: ", student_priority)
411                 curr_time += TQ.burst_time()
412                 TQ.dequeue()
413             elif tchr > stdnt:
414                 if curr_time >= tchr:
415                     student_priority += 1
416                     curr_time += TQ.burst_time()
417                     print("Teacher ", TQ.id_no()," issued book. Student Priority: ", student_priority)
418                     TQ.dequeue()
419                 else:
420                     curr_time += SQ.burst_time()
421                     print("Student ", SQ.id_no()," issued book")
422                     student_priority = 0
423                     SQ.dequeue()
424
425

```

```
IPython console
Console 1/A x
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

IPython console History log
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 than 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 3 rd Teacher at the counter and after that 2 more teacher arrives.						The student is served after the 4 th Teacher.	The student is served after 4 th Teacher.
12.	If 3 Student arrive first and a Teacher arrives at the same time with 2 nd Student at the counter and after that 4 more Student arrives.						The teacher is served after the 1 st student and the issuing goes on until the Student Queue is empty.	The teacher is served after the 1 st 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						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
	AT[T1] = 0 AT[T2] = 1 AT[T3] = 2		AT[S1] = 0 AT[S2] = 1					
14.	Total Teacher are 4 Total Student are 4						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
	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		

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

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