Date

M. Riyan Rasheed
F24- 186
Assignment 01

Task

Topic 02:
Event Planning System

Solution:

```
Class Agenda:
    def __init__(self):
        self.stack = []

    def push(self, session):
        self.stack.append(session)
        print(f"Submit: {session}")
    def pop(self):
        if not self.stack:
            print("No cancellation")
            return None
```

Date

```
remove = self.stack.pop
        print(f"Removed: {remove}")
        return removed

    def is_empty(self):
        return len(self.stack) == 0

    def display(self):
        if not self.stack:
            print("Empty")
        else:
            print("OK")
            for session in self.stack:
                print(session)
    def peak(self):
        if not self.stack:
            print("Empty")
            return None
            return self.stack[-1]
Class Admission:
    def __init__(self):
        self.queue = []
```

Date

```
def enqueue(self, attend):
    self.queue.append(attend)
    print(f"Enqueue: {attend}")

def dequeue(self):
    if not self.queue
        print("No waiting")
        return None
    Hadi = self.queue.pop(0)
    print(f"Admit: {attend Hadi}")
        return Hadi
def peak(self):
    if not self.queue:
        print("No attendee waiting")
        return None
        return self.queue[0]
def size(self):
    return len(self.queue)

def bubble(self):
    arr = self.queue
    n = len(arr
```

Date

```
for i in range(n):
    swape = false
    for j in range(0, n-1-j):
        if arr[j][i] < arr[j+1][i]:
            arr[j], arr[j+1] = arr[j+1]
            swap = True
        print(f"Pass {i}: {arr}")
            break
        self.queue = arr
```

Agenda Testing

```
DSA = st.Agenda()
DSA.push("Session 1: Keynote")
DSA.push("Session 2: AI Talk")
DSA.push("Session 3: Networking")
DSA.display()
DSA.pop()
DSA.display()
DSA.pop()
DSA.Display()
print("Agend {Empty}", DSA.is empty())
```

Date

Output

Submit : session 1 : Keynote
" : " 2 : AI Talk
Submit : session 3 : Networking
Current point : session 3 : Networking
Full agenda :
session 1 : Keynote
session 2 : AI Talk
session 3 : Networking
cancelled : session 3 : Networking
Full Agenda
session 1 : Keynote
Is Agenda Empty ? False

E.g of Queue

A = Admission()
A.enqueue("Riyan", 6)
A.enqueue("Haider", 9)
print("Before sorting:")
Print(A.queue)
A.BubbleU)

Date

```
print ("Sorted queue:")
    print (A.queue)
    print ('final:')
    while A.size()>0:
        A.dequeue()
```

Output

Enqueue: ('Riyan', b)
Enqueue: ('Haider', 9)
pass 1 [('Riyan', b), ('Haider', 9)]
pass 2 [('Haider', 9), ('Riyan', b)]
Sorted queue
Find admission queue
Admitted ('Haider', 9)
Remaing queue size 1
Admitted ('Riyan, b)
Remaining size:

Date

Task 01:

```
Class DocumentRevisionsystem:
    def __init__(self):
        self.revisions = []
    def add_revision(self, revision):
        self.revisions.append(revision)
        print(f"Added revision: {revision}")
    def undo_revision(self):
        if not self.revisions:
            print("No revisions")
            return None
        removed = self.revisions.pop()
        print(f"Undo: Removed {removed}")
        if self.revisions:
            print(f"Current revision: {self.revision}")
        else:
            print("No left")
        return removed

    def current_revision(self):
        if not self.revisions:
```

Date

```
    if not self.revisions:
        print("No revisions")
        return None
        return self.revision[-1]

Class printJobSystem:
    def __init__(self):
        self.jobs = []

    def add_print_jobs(self, priority):
        self.jobs.append(priority)
        print(f"Added print job {priority}")

    def process_print_job(self):
        if not self.jobs:
            print("No jobs")
            return None
        job = self.jobs.pop(0)
        print(f"processing job {job}")
        return job
```

Date

```
def sort_print_jobs(self):
    arr = self.jobs.copy()
    n = len(arr)
    for i in range(n):
        swapped = False
        for j in range(0, n-i-1):
            if arr[j] > arr[j+1]:
                arr[j], arr[j+1] = arr[j+1], arr[j]
                swapped = True
        print(f"After pass {i+1}: {arr}")
        if not swapped:
            break
    self.jobs = arr
```

E.g

```
A = DocumentRevisionSystem()
A.add_revision("Version 1: Initial draft")
A.add_revision("Version 2: Added chapter")
A.add_revision("Version 3: Fixed typos")
A.show_revision_history()
```

Date

```
A. undo_revision()
A. str undo_revision()

B = PrintJobSystem()
B.Add_print_job(5)
B.add_print_job(2)
B.add_print_job(8)

print("Before sorting:", print_system.jobs)
print_system.sort_print_jobs()
print("Sorted queue:", print_system.jobs)

while print_system.jobs:
    print_system.process_print_job()
```

Output

```
Added revision: Version1: intial draft
Added revision: Version2: Added chapter
Added revisions: Version3: fixed typos
Revision History
Version1: intial draft
Version2: Added chapter
Version3: fixed typos
```

Date

Undo: Removed version 3: fixed typos
Current revision: version 2: Added chapter
Revision History
version 1: Initial draft
version 2: Added chapter

Added print job with priority 5
" " " " " 2
" " " " " 4

After pass 1: [25, 8]
After pass 2: [2, 8, 5]
After pass 3: [8, 5, 2]

Sorted queue: (8, 8, 2)

Processing job with priority 8
" " " " 5
" " " " 2