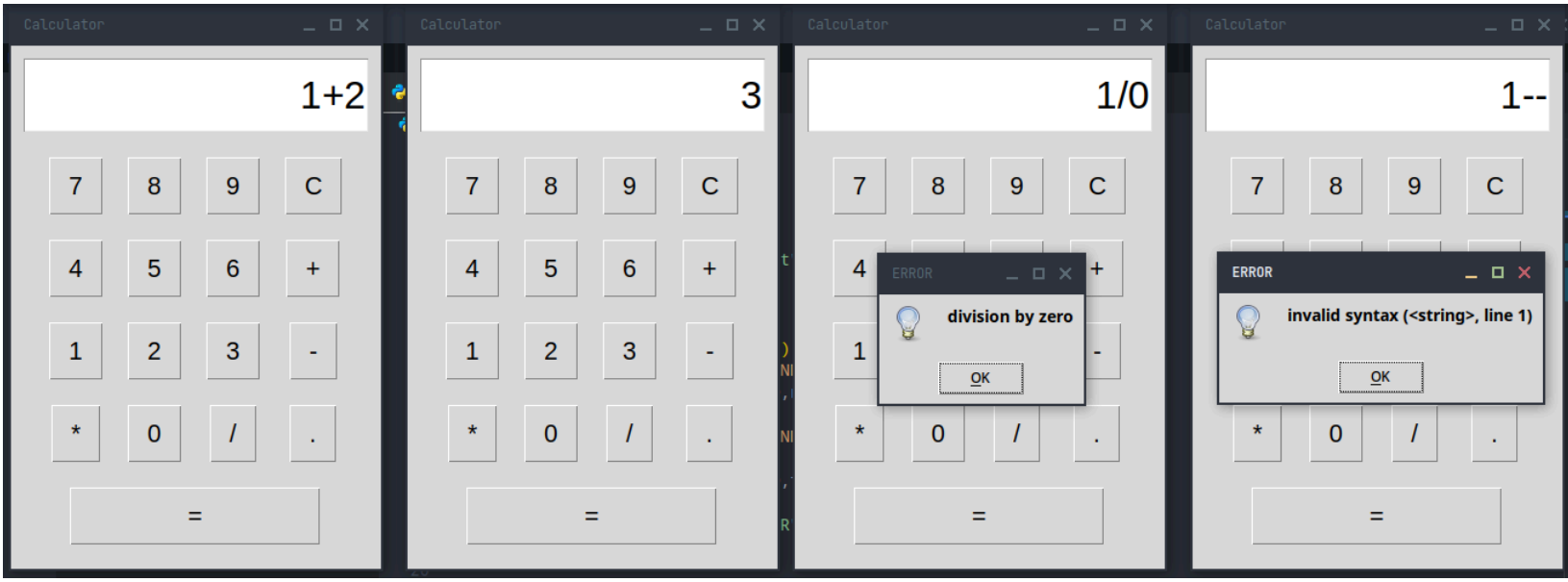
Question 1

Source Code:

```
import tkinter as tk
from tkinter import messagebox

def click(event):
   current = display.get()
   text = event.widget.cget("text")

   try:
       if text=="=":
           result = eval(current)
           display.delete(0,tk.END)
           display.insert(tk.END,result)
       elif text=="C":
           display.delete(0,tk.END)
       else:
           display.insert(tk.END,text)
   except Exception as e:
       messagebox.showinfo("ERROR",e)
       display.delete(0,tk.END)

window = tk.Tk()
window.title("Calculator")
window.geometry("320x450")

display = tk.Entry(window,font=("Arial",25),justify="right")
display.pack(fill=tk.X,padx=10,pady=10,ipady=10)

btn_frame = tk.Frame(window)
btn_frame.pack()

btn_labels = [
   ["7","8","9","C"],
   ["4","5","6","+"],
   ["1","2","3","-"],
   ["*","0","/","."],
   ["="]
]
```

```
for i in range(0,4):
   for j in range(0,4):
       button =
tk.Button(btn_frame,font=("Arial",16),padx=15,pady=10,text=btn_labels[i][j
])
       button.grid(row=i,column=j, padx=10,pady=10)
       button.bind("<Button>",click)

button =
tk.Button(btn_frame,font=("Arial",16),padx=100,pady=10,text=btn_labels[4][
0])
button.grid(row=5,column=0, columnspan=4, rowspan = 1, padx=10,pady=10)
button.bind("<Button>",click)

window.mainloop()
```

Output:



Question 2:

Source Code:

```
from PIL import Image, ImageFilter

def resize_image(image_path, width, height, output_path):

   image = Image.open(image_path)
   resized_image = image.resize((width, height))
   resized_image.save(output_path)

def rotate_image(image_path, angle, output_path):

   image = Image.open(image_path)
   rotated_image = image.rotate(angle)
   rotated_image.save(output_path)

def grayscale_image(image_path, output_path):

   image = Image.open(image_path)
   grayscale_image = image.convert("L")
   grayscale_image.save(output_path)

def filter_image(image_path, filter_type, output_path):

   image = Image.open(image_path)
   filtered_image = image.filter(filter_type)
   filtered_image.save(output_path)

def crop_image(image_path, bbox, output_path):

   image = Image.open(image_path)
   cropped_image = image.crop(bbox)
   cropped_image.save(output_path)

def main():
   image_path = "image.jpg"

   # Resize
   resize_image(image_path, 300, 200, "resized_image.jpg")
```

```
    # Rotate
    rotate_image(image_path, 90, "rotated_image.jpg")

    # Grayscale
    grayscale_image(image_path, "grayscale_image.jpg")

    # Filter
    filter_image(image_path, ImageFilter.BLUR, "blurred_image.jpg")

    # Crop
    crop_bbox = (100, 100, 400, 300)
    crop_image(image_path, crop_bbox, "cropped_image.jpg")

if __name__ == "__main__":
    main()
```

Output:

```
# Input Image
```

```
# Blurred Image
```

```
# Rotated Image
```

```
# Grayscale Image
```

```
# Resized Image
```

```
# Cropped Image
```

Question 3:

Source Code:

```python
class Time:
    def __init__(self, hours, minutes, seconds):
        self.hours = hours
        self.minutes = minutes
        self.seconds = seconds

    def __str__(self):
        return f"{self.hours:02d}:{self.minutes:02d}:{self.seconds:02d}"

    def normalize(self):
        extra_minutes, self.seconds = divmod(self.seconds, 60)
        self.minutes += extra_minutes
        extra_hours, self.minutes = divmod(self.minutes, 60)
        self.hours += extra_hours

    def __add__(self, other):
        total_hours = self.hours + other.hours
        total_minutes = self.minutes + other.minutes
        total_seconds = self.seconds + other.seconds

        result_time = Time(total_hours, total_minutes, total_seconds)
        result_time.normalize()
        return result_time

# Example usage
time1 = Time(1, 30, 45)
time2 = Time(2, 15, 20)

result_time = time1 + time2
print("Result:", result_time)
```

Output :

```
┌[jera@jera-acerone14z2493] [/dev/pts/3] [main ⚡]
└[~/python/python]> python3 addingTime.py
Result: 03:46:05
┌[jera@jera-acerone14z2493] [/dev/pts/3] [main ⚡]
└[~/python/python]>
```

Question 4:

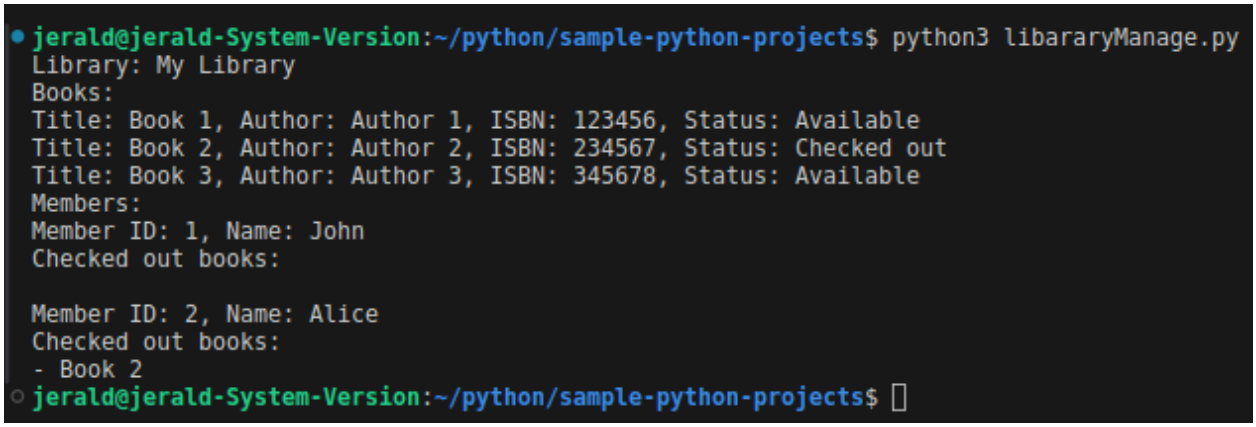
Source Code:

```
class Book:
    def __init__(self, title, author, isbn):
        self.title = title
        self.author = author
        self.isbn = isbn
        self.checked_out = False

    def check_out(self):
        self.checked_out = True

    def return_book(self):
        self.checked_out = False

    def __str__(self):
        status = "Checked out" if self.checked_out else "Available"
        return f"Title: {self.title}, Author: {self.author}, ISBN:
{self.isbn}, Status: {status}"

class Library:
    def __init__(self, name):
        self.name = name
        self.books = []
        self.members = []

    def add_book(self, book):
        self.books.append(book)

    def remove_book(self, isbn):
        for book in self.books:
            if book.isbn == isbn:
                self.books.remove(book)
                return

    def add_member(self, member):
        self.members.append(member)

    def remove_member(self, member_id):
```

```
        for member in self.members:
            if member.member_id == member_id:
                self.members.remove(member)
                return

    def checkout_book(self, isbn, member_id):
        for book in self.books:
            if book.isbn == isbn and not book.checked_out:
                book.check_out()
                for member in self.members:
                    if member.member_id == member_id:
                        member.check_out_book(book)
                        return

    def return_book(self, isbn, member_id):
        for member in self.members:
            if member.member_id == member_id:
                for book in member.checked_out_books:
                    if book.isbn == isbn:
                        book.return_book()
                        member.return_book(book)
                        return

    def __str__(self):
        book_list = "\n".join([str(book) for book in self.books])
        member_list = "\n".join([str(member) for member in self.members])
        return f"Library:
{self.name}\nBooks:\n{book_list}\nMembers:\n{member_list}"

class Member:
    def __init__(self, member_id, name):
        self.member_id = member_id
        self.name = name
        self.checked_out_books = []

    def check_out_book(self, book):
        self.checked_out_books.append(book)

    def return_book(self, book):
        self.checked_out_books.remove(book)
```

```
    def __str__(self):
        checked_out_books_str = "\n".join([f"- {book.title}" for book in
self.checked_out_books])
        return f"Member ID: {self.member_id}, Name: {self.name}\nChecked
out books:\n{checked_out_books_str}"


# Demo
if __name__ == "__main__":
    # Create books
    book1 = Book("Book 1", "Author 1", "123456")
    book2 = Book("Book 2", "Author 2", "234567")
    book3 = Book("Book 3", "Author 3", "345678")

    # Create members
    member1 = Member(1, "John")
    member2 = Member(2, "Alice")

    # Create library
    library = Library("My Library")

    # Add books and members to the library
    library.add_book(book1)
    library.add_book(book2)
    library.add_book(book3)
    library.add_member(member1)
    library.add_member(member2)

    # Checkout books
    library.checkout_book("123456", 1)
    library.checkout_book("234567", 2)

    # Return books
    library.return_book("123456", 1)

    # Display library status
    print(library)
```

Output:


jerald@jerald-System-Version:~/python/sample-python-projects$ python3 libararyManage.py
Library: My Library
Books:
Title: Book 1, Author: Author 1, ISBN: 123456, Status: Available
Title: Book 2, Author: Author 2, ISBN: 234567, Status: Checked out
Title: Book 3, Author: Author 3, ISBN: 345678, Status: Available
Members:
Member ID: 1, Name: John
Checked out books:
Member ID: 2, Name: Alice
Checked out books:
- Book 2
jerald@jerald-System-Version:~/python/sample-python-projects$

Question 5:

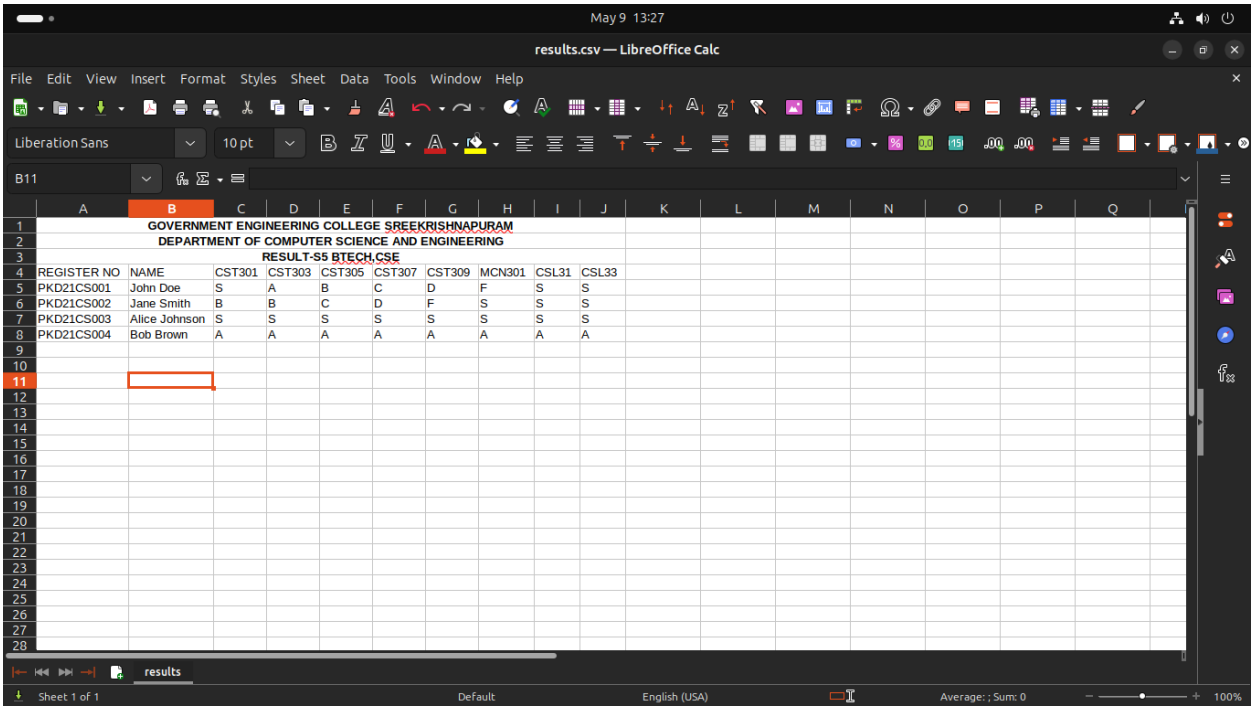
Source Code:

```
import pandas as pd

# Load data from CSV file
df = pd.read_csv('/home/jerald/python/sample-python-projects/results.csv',
skiprows=3)

# Display students with S grade in all subjects
students_with_s_grade = df[(df.iloc[:, 2:] == 'S').all(axis=1)][['REGISTER
NO', 'NAME']]
print("Students with S grade in all subjects:")
print(students_with_s_grade)

# Compute pass percentage for each subject
subject_pass_percentages = (df.iloc[:, 2:] != 'F').mean() * 100
print("\nPass percentage for each subject:")
print(subject_pass_percentages)

# Display students who have passed all subjects
students_passed_all_subjects = df[(df.iloc[:, 2:] !=
'F').all(axis=1)][['REGISTER NO', 'NAME']]
print("\nStudents who have passed all subjects:")
print(students_passed_all_subjects)
```

Output:

```
jerald@jerald-System-Version:~/python/sample-python-projects$ python3 csvFileManage.py
Students with S grade in all subjects:
  REGISTER NO           NAME
2  PKD21CS003  Alice Johnson

Pass percentage for each subject:
CST301    100.0
CST303    100.0
CST305    100.0
CST307    100.0
CST309     75.0
MCN301     75.0
CSL31     100.0
CSL33     100.0
dtype: float64

Students who have passed all subjects:
  REGISTER NO           NAME
2  PKD21CS003  Alice Johnson
3  PKD21CS004      Bob Brown
jerald@jerald-System-Version:~/python/sample-python-projects$
```

results.csv


May 9 13:27
results.csv — LibreOffice Calc
File Edit View Insert Format Styles Sheet Data Tools Window Help
Liberation Sans
10 pt
B11
GOVERNMENT ENGINEERING COLLEGE SREEKRISHNAPURAM
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
RESULT-S5 BTECH.CSE
REGISTER NO NAME CST301 CST303 CST305 CST307 CST309 MCN301 CSL31 CSL33
PKD21CS001 John Doe S A B C D F S S
PKD21CS002 Jane Smith B B C D F S S S
PKD21CS003 Alice Johnson S S S S S S S S
PKD21CS004 Bob Brown A A A A A A A A
results
Sheet 1 of 1
Default
English (USA)
Average: ; Sum: 0
100%