Student Name: Prem
UID: 24IMC12508
Branch: MCA
Section/Group: 1-A
Semester: 1st Sem
Date of Performance: 22-10-2024
Subject Name: DAA LAB
Subject Code: 24CAP-612

Aim/Overview of the practical:

1. Task to be done:

To Show Real Life Implementation of Quick Sort we will create top n student fetching system using quick sort:

Lets implement following features in our code to show the real life working of quick sort.

- Add student details (name and marks).
- Find and display the top N students who scored 90% or above, sorted in descending order of marks.
- Display all student data.
- Error handling for invalid inputs and empty student data.

This program allows users to manage student data in an interactive and visually appealing way, with built-in sorting and filtering for top-performing students.

2. Code for experiment/practical:

```
import tkinter as tk
from tkinter import messagebox

# Define the Student class
class Student:
    def __init__(self, name, marks):
        self.name = name
        self.marks = marks
```

```
    def __repr__(self):
        return f"{self.name}: {self.marks}"

# QuickSort Implementation
def quick_sort(arr, low, high):
    if low < high:
        pi = partition(arr, low, high)
        quick_sort(arr, low, pi - 1)
        quick_sort(arr, pi + 1, high)

def partition(arr, low, high):
    pivot = arr[high].marks  # Use marks as the pivot
    i = low - 1
    for j in range(low, high):
        if arr[j].marks >= pivot:  # Sorting in descending order
            i += 1
            arr[i], arr[j] = arr[j], arr[i]
    arr[i + 1], arr[high] = arr[high], arr[i + 1]
    return i + 1

# Function to find the top N students who scored 90% or above
def find_top_students():
    try:
        N = int(top_n_entry.get())
    except ValueError:
        messagebox.showerror("Invalid Input", "Please enter a valid number for
top students.")
        return
    if not students_list:
        messagebox.showinfo("No Data", "No student data available.")
        return
```

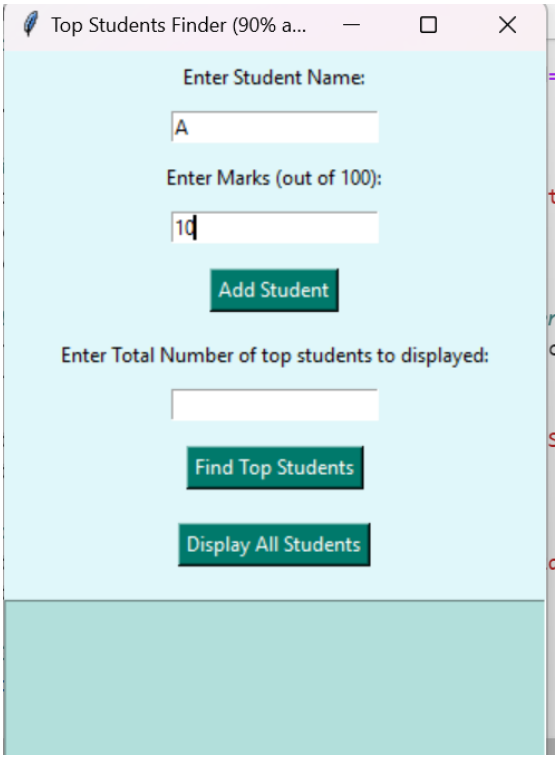
```
    # Filter students who scored 90% or above
    filtered_students = [student for student in students_list if student.marks >=
90]

    if not filtered_students:
        messagebox.showinfo("No Eligible Students", "No students have scored
90% or more.")
        return

    # Sort the students using QuickSort
    quick_sort(filtered_students, 0, len(filtered_students) - 1)

    # Get top N students
    top_students = filtered_students[:N] if N <= len(filtered_students) else
filtered_students

    # Clear the output area and display top students
    result_text.delete(1.0, tk.END)
    result_text.insert(tk.END, f"Top {N} Students (90% or above):\n\n")
    for student in top_students:
        result_text.insert(tk.END, f"{student.name}: {student.marks}%\n")

# Function to display the entire list of students
def display_all_students():
    if not students_list:
        messagebox.showinfo("No Data", "No student data available.")
        return
    # Clear the output area and display all students
    result_text.delete(1.0, tk.END)
    result_text.insert(tk.END, "All Students:\n\n")
    for student in students_list:
        result_text.insert(tk.END, f"{student.name}: {student.marks}%\n")
```

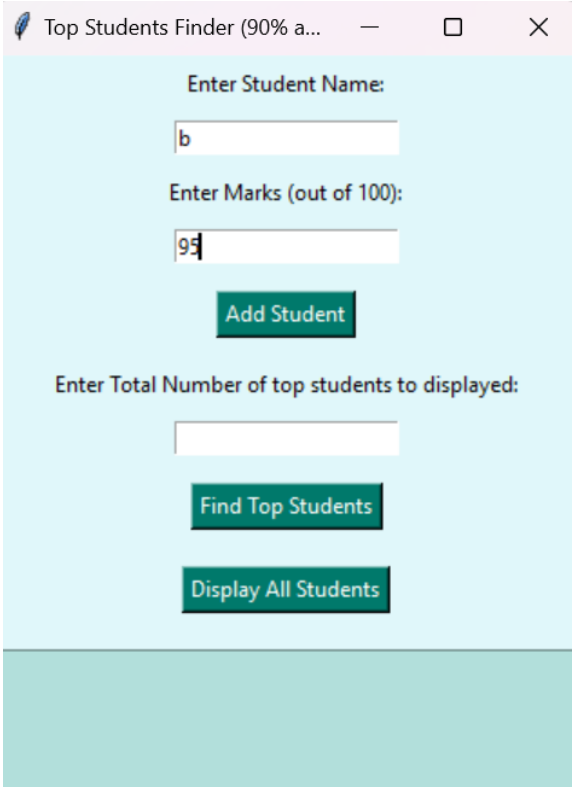
```
# Function to add student data
def add_student():
    name = name_entry.get()
    try:
        marks = float(marks_entry.get())
    except ValueError:
        messagebox.showerror("Invalid Marks", "Please enter a valid number for
marks.")
        return

    if name and marks >= 0:
        students_list.append(Student(name, marks))
        name_entry.delete(0, tk.END)
        marks_entry.delete(0, tk.END)
        messagebox.showinfo("Success", f"Student {name} added
successfully!")
    else:
        messagebox.showerror("Invalid Input", "Please enter a valid student
name and marks.")

# Create the main window
root = tk.Tk()
root.title("Top Students Finder (90% and Above)")
root.configure(bg='#e0f7fa')  # Light cyan background color

# List to store students
students_list = []

# Create input fields for student data
tk.Label(root, text="Enter Student Name:", bg='#e0f7fa').pack(pady=5)
name_entry = tk.Entry(root)
```

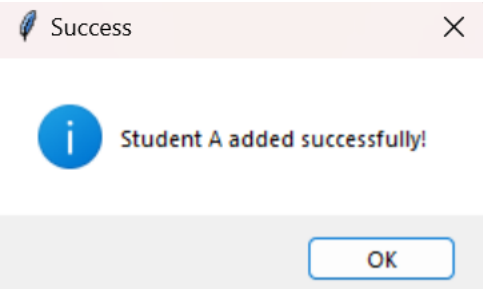
```
name_entry.pack(pady=5)

tk.Label(root, text="Enter Marks (out of 100):", bg='#e0f7fa').pack(pady=5)
marks_entry = tk.Entry(root)
marks_entry.pack(pady=5)

# Button to add student
add_button = tk.Button(root, text="Add Student", command=add_student,
bg='#00796b', fg='white')
add_button.pack(pady=10)

# Create field for user to enter number of top students
tk.Label(root, text="Enter Total Number of top students to displayed:",
bg='#e0f7fa').pack(pady=5)
top_n_entry = tk.Entry(root)
top_n_entry.pack(pady=5)

# Buttons to find and display top students and all students
find_button = tk.Button(root, text="Find Top Students",
command=find_top_students, bg='#00796b', fg='white')
find_button.pack(pady=10)

all_students_button = tk.Button(root, text="Display All Students",
command=display_all_students, bg='#00796b', fg='white')
all_students_button.pack(pady=10)

# Text area to display results
result_text = tk.Text(root, height=10, width=40, bg='#b2dfdb', fg='black')
result_text.pack(pady=10)

# Run the GUI loop
root.mainloop()
```

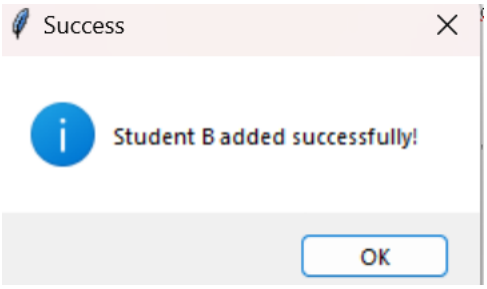
Output:

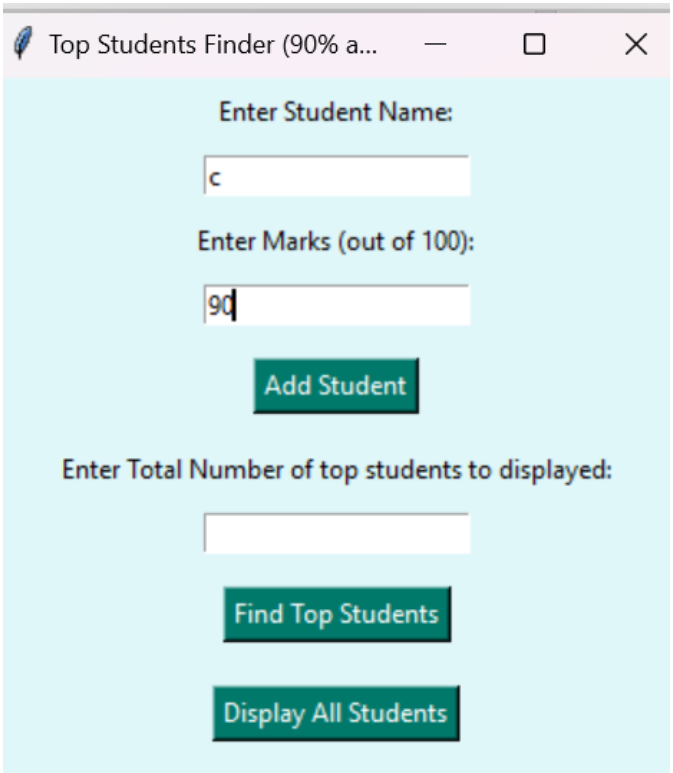
Adding students one by one:

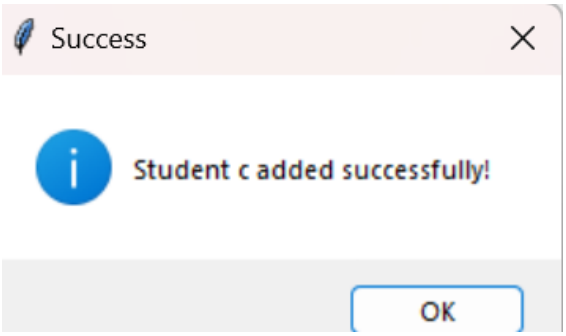


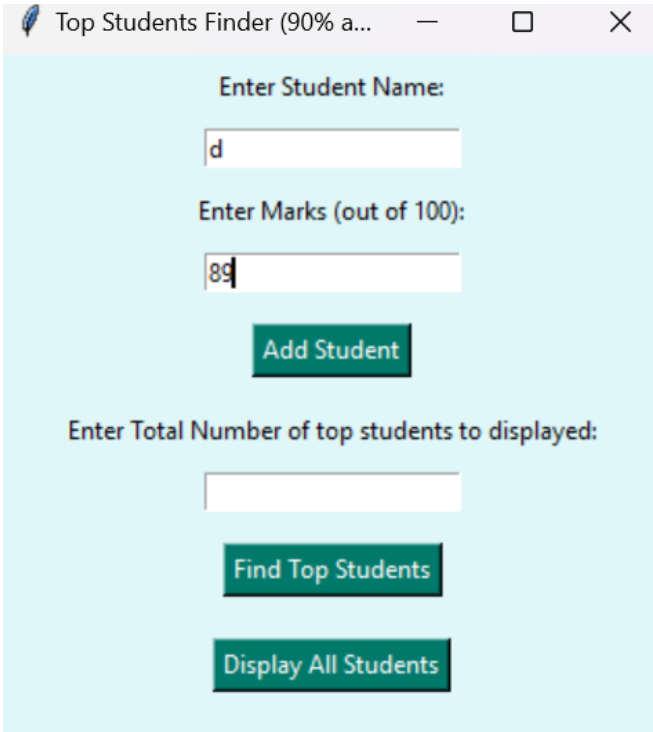


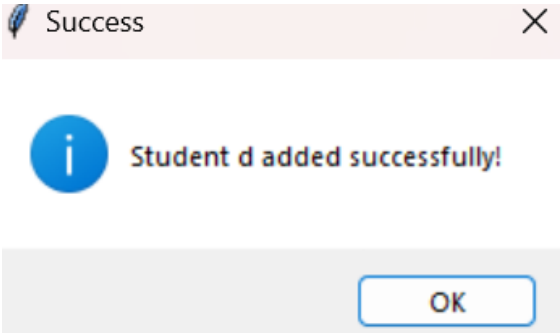




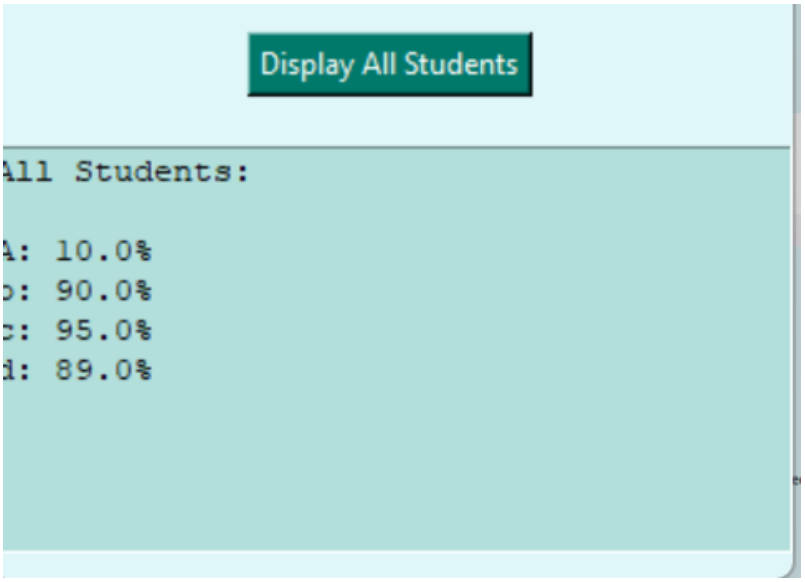

Top Students Finder (90% a...
Enter Student Name:
c
Enter Marks (out of 100):
90
Add Student
Enter Total Number of top students to displayed:
Find Top Students
Display All Students


Success
Student c added successfully!
OK

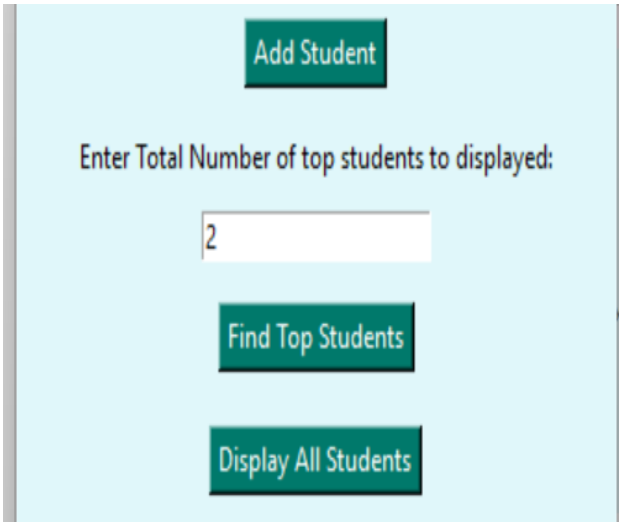

Top Students Finder (90% a...
Enter Student Name:
d
Enter Marks (out of 100):
89
Add Student
Enter Total Number of top students to displayed:
Find Top Students
Display All Students

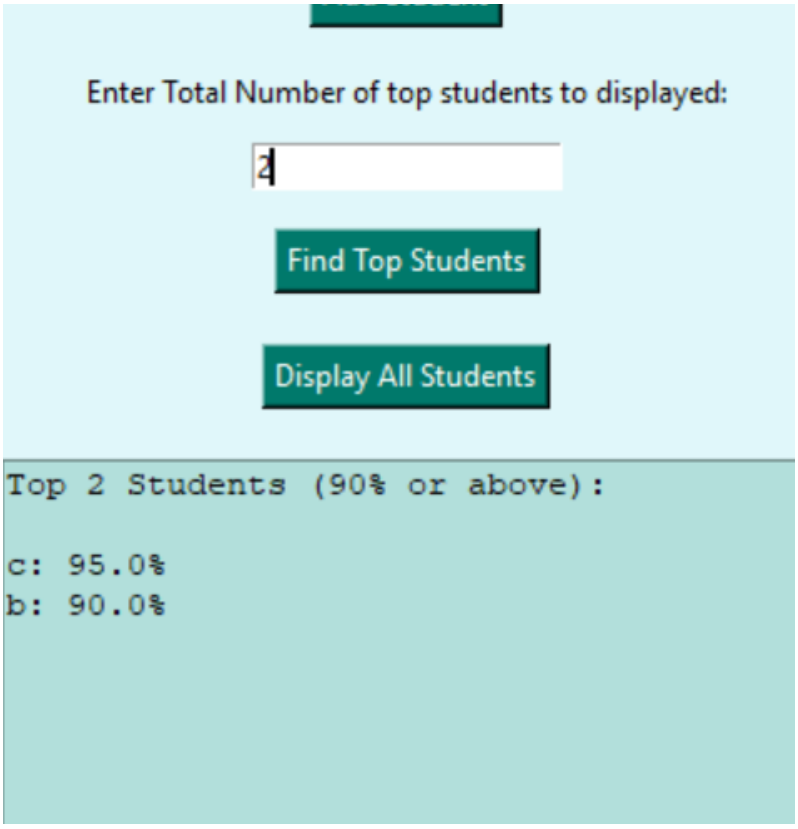

Success
Student d added successfully!
OK

DISPLAY ALL STUDENTS :



DISPLAT TOP 2 STUDENTS:





Learning Outcomes:

1. **Object-Oriented Programming (OOP):**
   - Understand how to create and use classes and objects.
   - Learn to define and use attributes and methods in a class.
2. **Sorting Algorithms:**
   - Implement the QuickSort algorithm to sort a list of objects based on their attributes.
   - Understand concepts like pivot selection, partitioning, and recursive sorting.
3. **Tkinter GUI Development:**
   - Learn how to create a basic GUI application using tkinter.
   - Handle user input with Entry fields, and show results with Text widgets.
4. **Error Handling and Input Validation:**
   - Use messageboxes for error handling and displaying feedback.
   - Validate user input (e.g., checking for numeric marks and valid student count).
5. **Data Structures:**
   - Work with lists to store and manipulate a collection of objects.
   - Learn how to filter and slice lists to extract specific data.
6. **Problem Solving:**
   - Break down a task into smaller parts: adding data, filtering, sorting, and displaying results.
   - Develop logical thinking skills while designing and implementing algorithms.