PIMPRI CHINCHWAD EDUCATION TRUST's.

PIMPRI CHINCHWAD COLLEGE OF ENGINEERING

(An Autonomous Institute)

S.Y. B. TECH **Year:** 2024 – 25 **Semester:** I

Name: Sonawane Prachi Mahendra. **PRN:** 124B2B018

Department : Computer Engineering **Division:** B

Course : Data Structures Laboratory **Course Code:** BCE23PC02

Date: 9/08/24

Assignment – 1

- **Aim:**

  Consider a student database of SY COMP class (at least 15 records). Database contains different fields of every student like Roll No, Name and SGPA.

  1. Design a roll call list, arrange list of students according to roll numbers in ascending order using Insertion Sort.
  2. Arrange list of students alphabetically using shell sort.
  3. Arrange list of students to find out first ten toppers from a class using Radix sort

- **Source Code:**

  1. Design a roll call list, arrange list of students according to roll numbers in ascending order using Insertion Sort:

```
#include <iostream> #include

<string>


struct Student

{

   int rollNo; std::string name;

   float sgpa;
```

```
};
void insertionSort(Student students[], int n)
{
   for (int i = 1; i < n; i++)
   {
      Student key = students[i]; int j = i -
      1;
      while (j >= 0 && students[j].rollNo > key.rollNo)
      {
         students[j + 1] = students[j]; j--;
      }
      students[j + 1] = key;
   }
}
void printStudents(const Student students[], int n)
{
   for (int i = 0; i < n; i++)
   {
      std::cout << "Roll No: " << students[i].rollNo<< ", Name:" << students[i].name
<< ", SGPA: " << students[i].sgpa << std::endl;
   }
}
int main()
{
   Student students[15] = { {10, "Alice", 8.5}, {5, "Bob", 9.1}, {3, "Charlie", 7.2},
```

```
{12, "David", 8.9}, {1, "Eve", 9.5}, {7, "Frank", 6.4}, {9, "Grace", 8.1}, {2,
"Hannah", 7.8}, {15, "Ivy", 6.9}, {8, "Jack", 7.5}, {11, "Karen", 8.7}, {6, "Leo", 9.3},
{4, "Mona", 8.0}, {14, "Nina", 9.4}, {13, "Oscar", 7.1} };

    insertionSort(students, 15);

    printStudents(students, 15);

    return 0;

}
```

2. Arrange list of students alphabetically using shell sort:

```
#include <iostream>
#include <string>
struct Student
{
   int rollNo; std::string name;
   float sgpa;
};
void shellSort(Student students[], int n)
{
   for (int gap = n / 2; gap > 0; gap /= 2)
   {
      for (int i = gap; i < n; i++)
      {
         Student temp = students[i]; int j;

         for (j = i; j >= gap && students[j - gap].name > temp.name; j -= gap)
         {
            students[j] = students[j - gap];
         }
         students[j] = temp;
      }
   }
}
```

```
void printStudents(const Student students[], int n)
{
    for (int i = 0; i < n; i++)
    {
        std::cout << "Roll No: " << students[i].rollNo
        << ", Name: " << students[i].name
        << ", SGPA: " << students[i].sgpa << std::endl;
    }
}
int main()
{
    Student students[15] =
    {
        {10, "Alice", 8.5}, {5, "Bob", 9.1}, {3, "Charlie", 7.2},
        {12, "David", 8.9}, {1, "Eve", 9.5}, {7, "Frank", 6.4},
        {9, "Grace", 8.1}, {2, "Hannah", 7.8}, {15, "Ivy", 6.9},
        {8, "Jack", 7.5}, {11, "Karen", 8.7}, {6, "Leo", 9.3},
        {4, "Mona", 8.0}, {14, "Nina", 9.4}, {13, "Oscar", 7.1}
    };
    shellSort(students, 15);
    printStudents(students, 15);
    return 0;
}
```

3. Arrange list of students to find out first ten toppers from a class using Radix sort:

```
#include <iostream>

#include <string>

using namespace std;
struct Student
{
   int rollNo; string name;
   float sgpa;
};
void radixSort(Student students[], int n)
{
   int max = students[0].rollNo; for (int i
   = 1; i < n; i++)
   {
      if (students[i].rollNo > max)
      {
         max = students[i].rollNo;
      }
   }
   for (int exp = 1; max / exp > 0; exp *= 10)
   {
      Student output[n]; int
      count[10] = {0};
```

```
        for (int i = 0; i < n; i++)
        {
            count[(students[i].rollNo / exp) % 10]++;
        }
        for (int i = 1; i < 10; i++)
        {
            count[i] += count[i - 1];
        }
        for (int i = n - 1; i >= 0; i--)
        {
            output[count[(students[i].rollNo / exp) % 10] - 1] = students[i];
            count[(students[i].rollNo / exp) % 10]--;
        }
        for (int i = 0; i < n; i++)
        {
            students[i] = output[i];
        }
    }
}
void sortStudentsBySGPA(Student students[], int n)
{
    for (int i = 0; i < n - 1; i++)
    {
        for (int j = i + 1; j < n; j++)
```

```
        {
            if (students[i].sgpa < students[j].sgpa)
            {
                Student temp = students[i]; students[i]
                = students[j]; students[j] = temp;
            }
        }
    }
}
int main() {
        Student students[] = {
            {1, "Rahul", 8.5},
            {2, "Priya", 9.2},
        {3, "Rohan", 8.8},
        {4, "Aisha", 7.9},
        {5, "Karan", 9.5},
        {6, "Sonia", 8.2},
        {7, "Amit", 7.5},
        {8, "Neha", 9.0},
        {9, "Vikram", 8.6},
        {10, "Shreya", 9.1},
        {11, "Gaurav", 8.1},
        {12, "Rucha", 7.8},
        {13, "Siddharth", 9.3},
```

```
        {14, "Tanvi", 8.4},
        {15, "Abhishek", 9.4},
    };
    int n = sizeof(students) / sizeof(students[0]);
    radixSort(students, n);
    sortStudentsBySGPA(students, n);

    cout << "Top 10 Toppers:" << endl; for (int i
    = 0; i < 10; i++)

cout << "Roll No: " << students[i].rollNo << ", Name: " << students[i].name << ", SGPA: "
<< students[i].sgpa << endl;

    return 0;
}
```

- **Screen shots of Output:**

1.

```
Output

/tmp/Y1kRQNEVCH.o
Roll No: 1, Name:Eve, SGPA: 9.5
Roll No: 2, Name:Hannah, SGPA: 7.8
Roll No: 3, Name:Charlie, SGPA: 7.2
Roll No: 4, Name:Mona, SGPA: 8
Roll No: 5, Name:Bob, SGPA: 9.1
Roll No: 6, Name:Leo, SGPA: 9.3
Roll No: 7, Name:Frank, SGPA: 6.4
Roll No: 8, Name:Jack, SGPA: 7.5
Roll No: 9, Name:Grace, SGPA: 8.1
Roll No: 10, Name:Alice, SGPA: 8.5
Roll No: 11, Name:Karen, SGPA: 8.7
Roll No: 12, Name:David, SGPA: 8.9
Roll No: 13, Name:Oscar, SGPA: 7.1
Roll No: 14, Name:Nina, SGPA: 9.4
Roll No: 15, Name:Ivy, SGPA: 6.9


=== Code Execution Successful ===
```

2.

```
Output

/tmp/NTaMXSCkVn.o
Top 10 Toppers:
Roll No: 5, Name: Karan, SGPA: 9.5
Roll No: 15, Name: Abhishek, SGPA: 9.4
Roll No: 13, Name: Siddharth, SGPA: 9.3
Roll No: 2, Name: Priya, SGPA: 9.2
Roll No: 10, Name: Shreya, SGPA: 9.1
Roll No: 8, Name: Neha, SGPA: 9
Roll No: 3, Name: Rohan, SGPA: 8.8
Roll No: 9, Name: Vikram, SGPA: 8.6
Roll No: 1, Name: Rahul, SGPA: 8.5
Roll No: 14, Name: Tanvi, SGPA: 8.4


=== Code Execution Successful ===
```

3.

```
Output

/tmp/23kSiUIDcJ.o
Roll No: 10, Name: Alice, SGPA: 8.5
Roll No: 5, Name: Bob, SGPA: 9.1
Roll No: 3, Name: Charlie, SGPA: 7.2
Roll No: 12, Name: David, SGPA: 8.9
Roll No: 1, Name: Eve, SGPA: 9.5
Roll No: 7, Name: Frank, SGPA: 6.4
Roll No: 9, Name: Grace, SGPA: 8.1
Roll No: 2, Name: Hannah, SGPA: 7.8
Roll No: 15, Name: Ivy, SGPA: 6.9
Roll No: 8, Name: Jack, SGPA: 7.5
Roll No: 11, Name: Karen, SGPA: 8.7
Roll No: 6, Name: Leo, SGPA: 9.3
Roll No: 4, Name: Mona, SGPA: 8
Roll No: 14, Name: Nina, SGPA: 9.4
Roll No: 13, Name: Oscar, SGPA: 7.1


=== Code Execution Successful ===
```

- **Conclusion:**

  Hence, we studied about various sorting techniques such as Insertion Sort, Shell Sort and Radix Sort with their Programs.