PIMPRI CHINCHWAD EDUCATION TRUST's.

PIMPRI CHINCHWAD COLLEGE OF ENGINEERING

(An Autonomous Institute)

S.Y. B. TECH **Year:** 2024 – 25 **Semester:** I
Name: Abhishek Joshi **PRN: 123B1B150**
Department: Computer Engineering **Division:** C (C1)
Course: Data Structures Laboratory **Course Code:** BCE23PC02
Date:

Assignment –2

- **Aim:**

Consider Employee database of PCCOE (at least 20 records). Database contains different fields of every employee like EMP-ID, EMP-Name and EMP-Salary.

a) Arrange list of employees according to EMP-ID in ascending order using Quick Sort
b) Arrange list of Employee alphabetically using Merge Sort

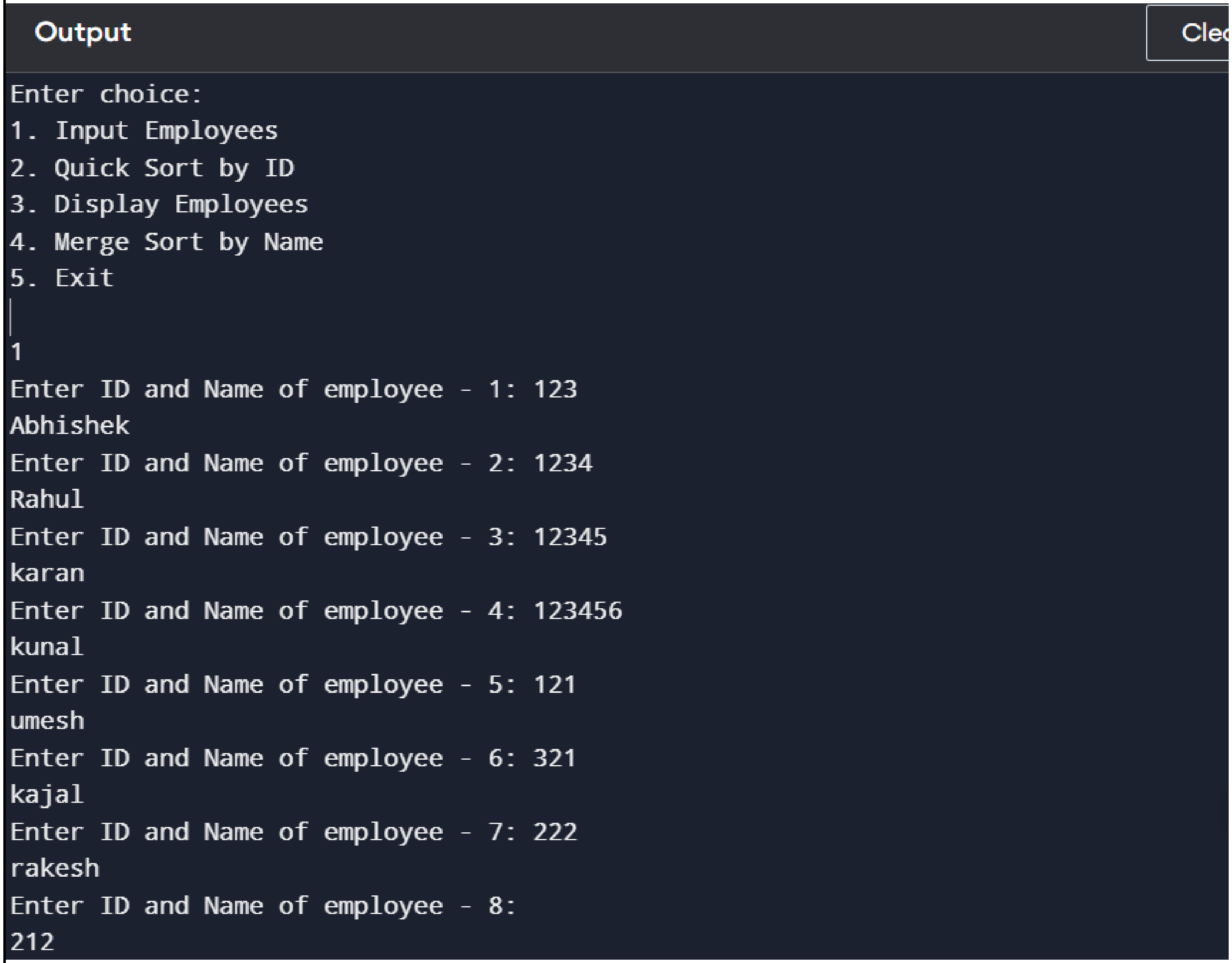
- **Source Code :**
  - Arrange list of employees according to EMP-ID in ascending order using Quick Sort

```
#include<iostream>
using namespace std;

class Employee{
   private:
      string name;
      int id;
   public:
      void input(Employee e[], int F, int L){
         for(int i=0; i<(L+1); ++i){
            cout<<"Enter Data of employee - "<<(i+1)<<": ";
            cin>>e[i].id>>e[i].name;
         }
      }
      void quickSort(Employee e[], int F, int L){
         if(F<L){
            int pivot = F;
            int i = F+1;
            int j = L;
            while(i<j){
                while(i <= L && e[i].id < e[pivot].id){
                    i++;
                }
                while(j >= F && e[j].id > e[pivot].id){
                    j--;
                }
                if(i<j){
                    Employee t;
                    t = e[i];
                    e[i] = e[j];
                    e[j] = t;
                }
                else{
                    break;
                }
            }
            Employee t1;
            t1 = e[j];
            e[j] = e[pivot];
            e[pivot] = t1;

        quickSort(e,F,j-1);
        quickSort(e,j+1,L);
        }
    }
void merge(Employee e[], int left, int mid, int right) {
    int n1 = mid - left + 1;
    int n2 = right - mid;
    Employee* L = new Employee[n1];
    Employee* R = new Employee[n2];

    for (int i = 0; i < n1; i++)
        L[i] = e[left + i];
    for (int j = 0; j < n2; j++)
        R[j] = e[mid + 1 + j];

    int i = 0, j = 0, k = left;
    while (i < n1 && j < n2) {
        if (L[i].name < R[j].name) {
            e[k] = L[i];
            i++;
        } else {
            e[k] = R[j];
```

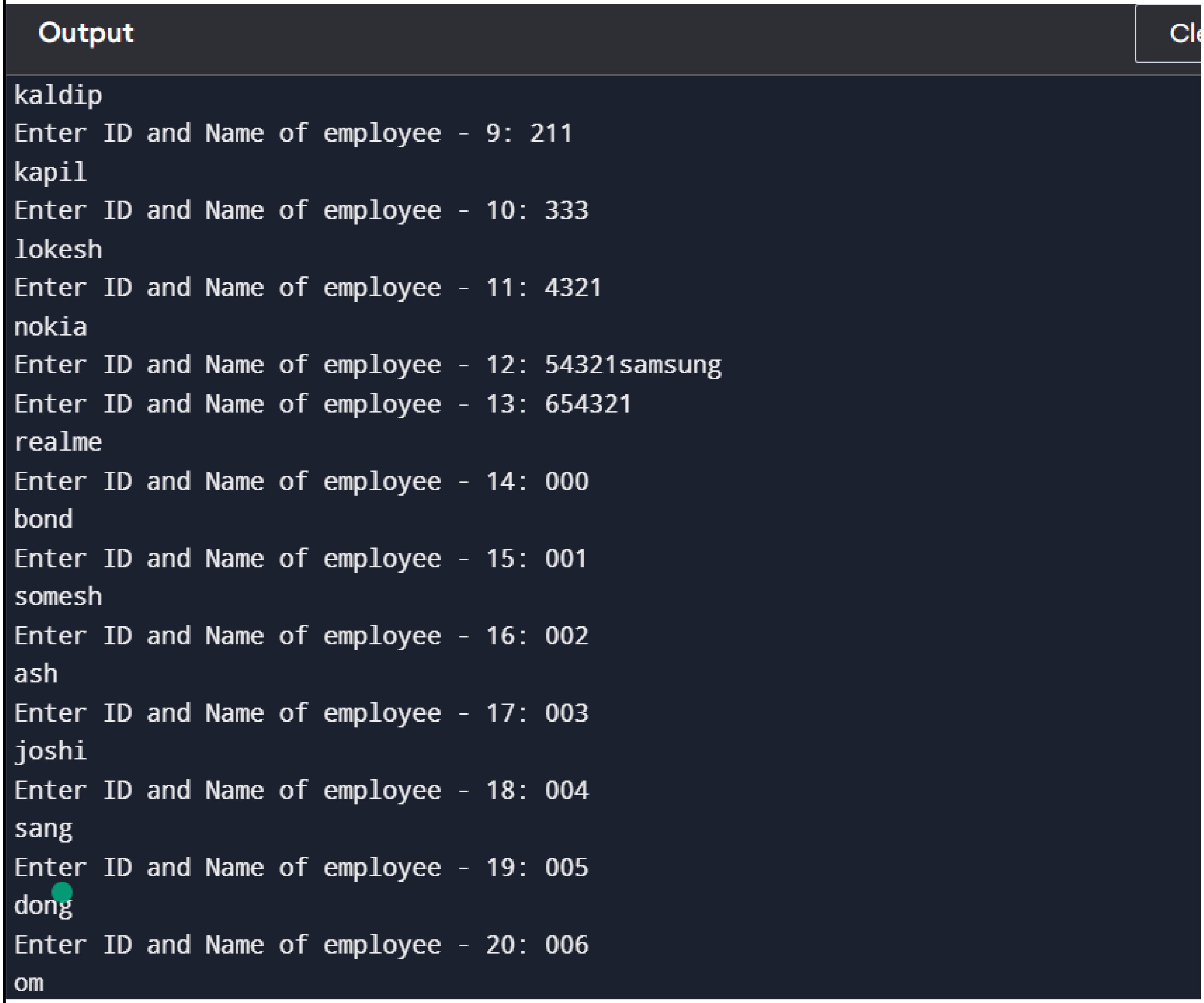
```
                j++;
            }
            k++;
        }

        while (i < n1) {
            e[k] = L[i];
            i++;
            k++;
        }

        while (j < n2) {
            e[k] = R[j];
            j++;
            k++;
        }

        delete[] L;
        delete[] R;
    }
    void mergeSort(Employee e[], int left, int right) {
        if (left < right) {
            int mid = left + (right - left) / 2;

            mergeSort(e, left, mid);
            mergeSort(e, mid + 1, right);
            merge(e, left, mid, right);
        }
    }
    void display(Employee e[], int F, int L){
            for(int i=0; i<(L+1); ++i){
                cout<<"Data of employee - "<<(i+1)<<": "<<e[i].id<<" "<<e[i].name<<endl;
                }
        }
};

int main(){
    Employee e[5], x;
    int n=sizeof(e)/ sizeof(e[0]);
    int F=0;
    int L=(n-1);
   int choice;
   do{
        cout<<"Enter choice: "; cin>>choice;
```

```
          switch(choice){
            case 1:
              x.input(e,F,L); break;
            case 2:
              x.quickSort(e,F,L); break;
            case 3:
              x.display(e,F,L); break;
            case 4:
              x.mergeSort(e, 0, n - 1); break;
            case 5:
              cout<<"Exit";
            default:
              break;
          }
        } while(choice != 5);
    }
```

- **<u>Screen Shot of Output :</u>**

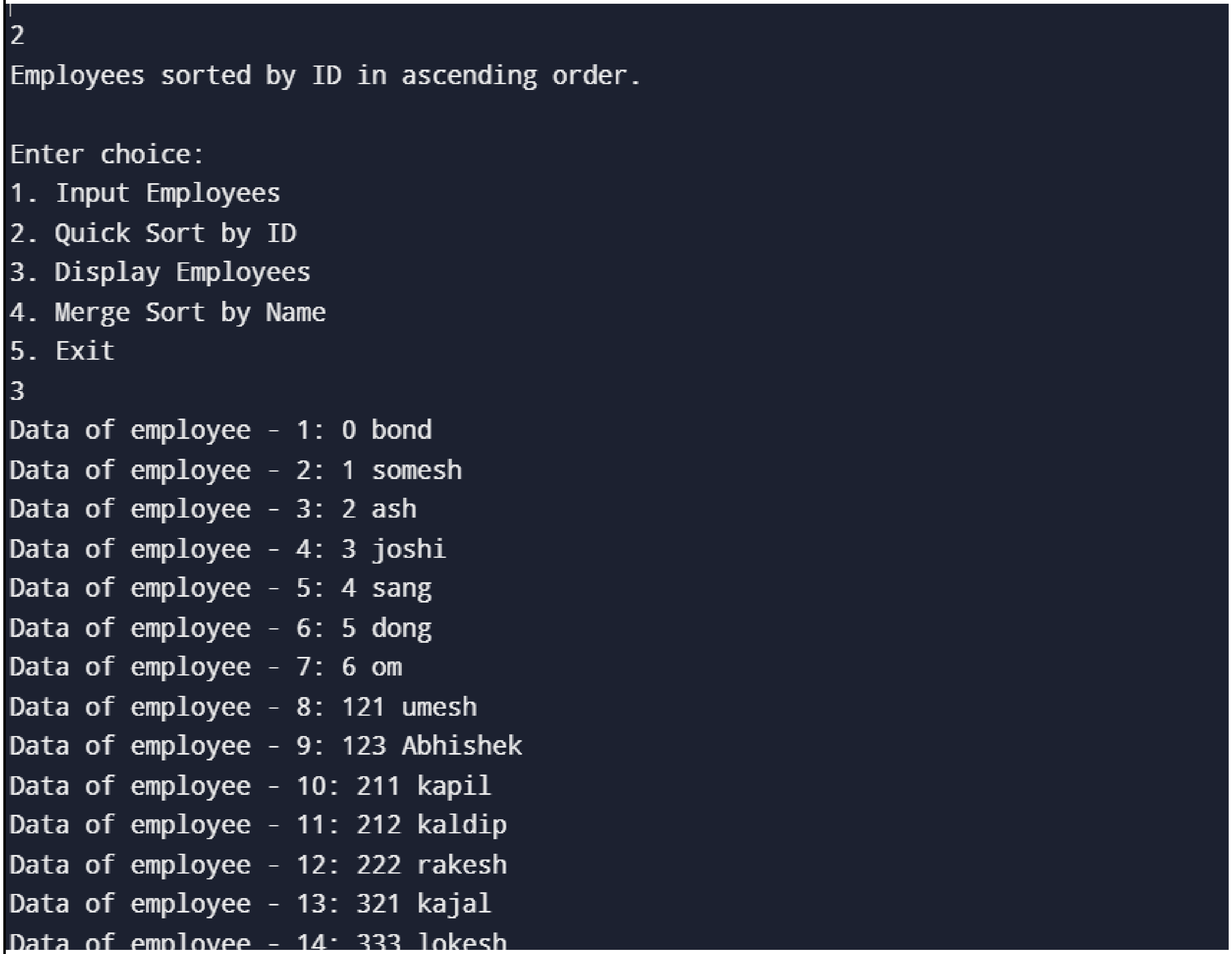


```
Enter choice:
1. Input Employees
2. Quick Sort by ID
3. Display Employees
4. Merge Sort by Name
5. Exit

1
Enter ID and Name of employee - 1: 123
Abhishek
Enter ID and Name of employee - 2: 1234
Rahul
Enter ID and Name of employee - 3: 12345
karan
Enter ID and Name of employee - 4: 123456
kunal
Enter ID and Name of employee - 5: 121
umesh
Enter ID and Name of employee - 6: 321
kajal
Enter ID and Name of employee - 7: 222
rakesh
Enter ID and Name of employee - 8:
212
```

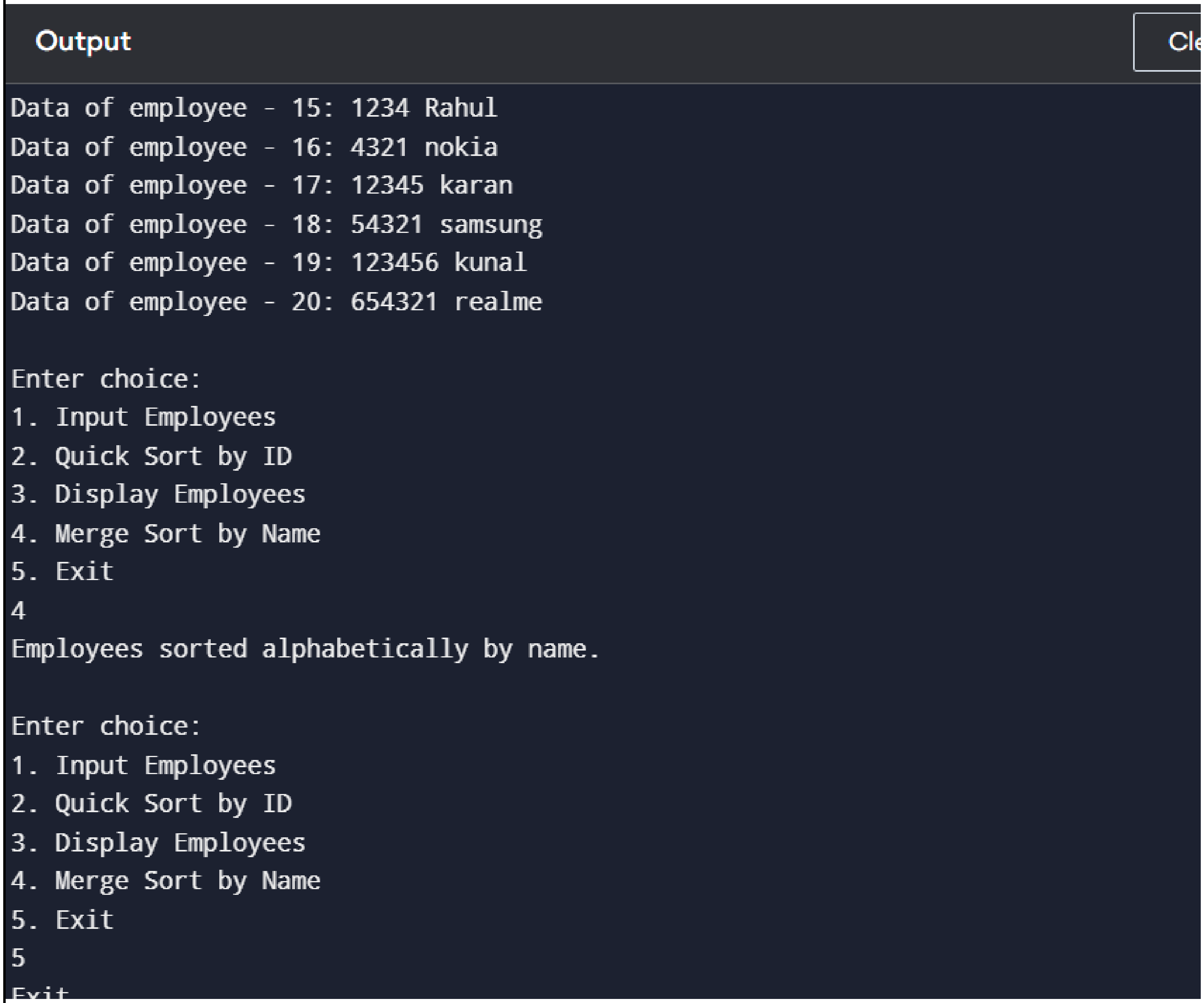

Output

Cl

```
kaldip
Enter ID and Name of employee - 9: 211
kapil
Enter ID and Name of employee - 10: 333
lokesh
Enter ID and Name of employee - 11: 4321
nokia
Enter ID and Name of employee - 12: 54321samsung
Enter ID and Name of employee - 13: 654321
realme
Enter ID and Name of employee - 14: 000
bond
Enter ID and Name of employee - 15: 001
somesh
Enter ID and Name of employee - 16: 002
ash
Enter ID and Name of employee - 17: 003
joshi
Enter ID and Name of employee - 18: 004
sang
Enter ID and Name of employee - 19: 005
dong
Enter ID and Name of employee - 20: 006
om
```



```
2
Employees sorted by ID in ascending order.

Enter choice:
1. Input Employees
2. Quick Sort by ID
3. Display Employees
4. Merge Sort by Name
5. Exit
3
Data of employee - 1: 0 bond
Data of employee - 2: 1 somesh
Data of employee - 3: 2 ash
Data of employee - 4: 3 joshi
Data of employee - 5: 4 sang
Data of employee - 6: 5 dong
Data of employee - 7: 6 om
Data of employee - 8: 121 umesh
Data of employee - 9: 123 Abhishek
Data of employee - 10: 211 kapil
Data of employee - 11: 212 kaldip
Data of employee - 12: 222 rakesh
Data of employee - 13: 321 kajal
Data of employee - 14: 333 lokesh
```



```
Data of employee - 15: 1234 Rahul
Data of employee - 16: 4321 nokia
Data of employee - 17: 12345 karan
Data of employee - 18: 54321 samsung
Data of employee - 19: 123456 kunal
Data of employee - 20: 654321 realme

Enter choice:
1. Input Employees
2. Quick Sort by ID
3. Display Employees
4. Merge Sort by Name
5. Exit
4
Employees sorted alphabetically by name.

Enter choice:
1. Input Employees
2. Quick Sort by ID
3. Display Employees
4. Merge Sort by Name
5. Exit
5
Exit
```

- **Conclusion:**
  Hence, we studied about various sorting techniques such as Quick Sort and Merge Sort with their algorithm and programs.