

Ganpat University
Faculty of Engineering & Technology
Computer Science & Engineering
Name:- Dwij Vatsal Desai
Sem:- 2
Sub: - ESFP-II
Enrollment No.:- 23162121027
Prac:- 8
Date:- 16/4/2024
Practical 8

Definition:

Complete the code for the object already assigned to you during practical 7 to satisfy following specifications.

This is the extended part of practical number-7, which is to be performed as per the following instructions :

1. Append all string data to first string of your object and then display only string data of all objects together (one record in one line).
2. Find the length of the string prepared in point 1, for all records and display on screen with appropriate message.
3. Display last number of characters from given string field like customer name, product name or brand, name, etc as per user choice.
4. Find first and last character of all the string fields found in point 1.
5. Extract substring from given string as per your string field available in the given record.
6. Design a search method, which asks a data to be searched based on name and other parameters of the object. Display record data if match found. Display appropriate message if match not found.
7. Display all record information/data record in ascending and descending orders based on their name field.

[Note: You must implement object-oriented concept for the practical]

Code:-

```
#include <iostream>
#include <fstream>
```

```

#include <string>

using namespace std;

void userinfo() {
    string age, name, address, email;

    cout << "Enter your info here (Age, Name, Address, Email): ";
    cin >> age >> name >> address >> email;
    cout << endl;

    // Append all strings into one
    string tinfo = age + " " + name + " " + address + " " + email;
    cout << "Concatenated string: " << tinfo << endl;

    // Display length of the concatenated string
    cout << "Length of concatenated string: " << tinfo.length() << endl;

    // Display last characters based on user choice
    int num_chars;
    cout << "Enter the number of characters to display from the end: ";
    cin >> num_chars;
    cout << "Last " << num_chars << " characters: " <<
tinfo.substr(tinfo.length() - num_chars) << endl;

    // Find and display first and last characters
    cout << "First character: " << tinfo.front() << endl;
    cout << "Last character: " << tinfo.back() << endl;

    // Extract substring based on user input
    int start_index, length;
    cout << "Enter the starting index and length of the substring to extract:
";
    cin >> start_index >> length;
    cout << "Extracted substring: " << tinfo.substr(start_index, length) <<
endl;

    // Search method
    string search_term;
    cout << "Enter the search term: ";
    cin >> search_term;
    if (tinfo.find(search_term) != string::npos) {
        cout << "Match found!" << endl;
    } else {
        cout << "No match found." << endl;
    }
}

```

```

class elearning {
private:
    int expression;
    string selectedCourse;
    string courseLevel;
    string preferredWayOfLearning;
    string durationForEachDay;

public:
    elearning() {
        expression = 0;
    }

    elearning(int exp) {
        expression = exp;
        switch (expression) {
            case 1:
                cout << endl << "You chose Business" << endl << endl;
                selectedCourse = "Business";
                selectComputerCourse();
                break;
            case 2:
                cout << endl << "You chose Computer course" << endl << endl;
                selectedCourse = "Computer course";
                selectComputerCourse();
                break;
            case 3:
                cout << endl << "You chose Data science" << endl << endl;
                selectedCourse = "Data science";
                selectComputerCourse();
                break;
            case 4:
                cout << endl << "You chose ICT" << endl << endl;
                selectedCourse = "ICT";
                selectComputerCourse();
                break;
            default:
                break;
        }
    }

    void selectCourse() {
        cout << ">>>Course Selection<<<" << endl;
        cout << "<1> Business" << endl;
        cout << "<2> Computer course" << endl;
        cout << "<3> Data science" << endl;
        cout << "<4> ICT" << endl;
    }
}

```

```

void selectComputerCourse() {
    int totalDuration = 0;
    int innerExpression = 0;
    while (true) {
        cout << endl << ">>>Enter choice for Computer course:<<<" << endl;
        cout << "<1> Course Level" << endl;
        cout << "<2> Preferred Way of Learning" << endl;
        cout << "<3> Duration for Each Day" << endl;
        cout << "<4> Exit" << endl;
        cin >> innerExpression;

        switch (innerExpression) {
            case 1: {
                int level = 0;
                cout << "Course Level:" << endl;
                cout << "<1> 1-2 weeks (basic)" << endl;
                cout << "<2> 2-5 weeks (intermediate)" << endl;
                cout << "<3> 6-7 weeks (advance)" << endl;
                cin >> level;
                switch (level) {
                    case 1:
                        cout << "1-2 weeks (basic)" << endl;
                        courseLevel = "1-2 weeks (basic)";
                        totalDuration += 2;
                        break;
                    case 2:
                        cout << "2-5 weeks (intermediate)" << endl;
                        courseLevel = "2-5 weeks (intermediate)";
                        totalDuration += 5;
                        break;
                    case 3:
                        cout << "6-7 weeks (advance)" << endl;
                        courseLevel = "6-7 weeks (advance)";
                        totalDuration += 7;
                        break;
                    default:
                        cout << "Invalid choice" << endl;
                        break;
                }
                break;
            }
            case 2: {
                int preferredWay = 0;
                cout << "Preferred Way of Learning:" << endl;
                cout << "<1> Guided project" << endl;
                cout << "<2> Personal practical" << endl;
                cout << "<3> Notes and concept learning" << endl;

```

```

        cin >> preferredWay;
        switch (preferredWay) {
        case 1:
            cout << "Guided project" << endl;
            preferredWayOfLearning = "Guided project";
            break;
        case 2:
            cout << "Personal practical" << endl;
            preferredWayOfLearning = "Personal practical";
            break;
        case 3:
            cout << "Notes and concept learning" << endl;
            preferredWayOfLearning = "Notes and concept learning";
            break;
        default:
            cout << "Invalid choice" << endl;
            break;
        }
        break;
    }
    case 3: {
        int duration = 0;
        cout << "Duration for Each Day:" << endl;
        cout << "<1> 2 hours" << endl;
        cout << "<2> 4 hours" << endl;
        cout << "<3> 6 hours" << endl;
        cin >> duration;
        switch (duration) {
        case 1:
            cout << "2 hours" << endl;
            durationForEachDay = "2 hours";
            break;
        case 2:
            cout << "4 hours" << endl;
            durationForEachDay = "4 hours";
            break;
        case 3:
            cout << "6 hours" << endl;
            durationForEachDay = "6 hours";
            break;
        default:
            cout << "Invalid choice" << endl;
            break;
        }
        break;
    }
    case 4:
        cout << "Exiting..." << endl;

```

```

        cout << "Total duration to complete the course: " <<
totalDuration << " weeks" << endl;
        storeDataToFile();
        return;
    default:
        cout << "Invalid choice" << endl;
        break;
    }
}

}

void storeDataToFile() {
    ofstream file("elearning_data.txt", ios_base::app);
    if (file.is_open()) {
        file << "Selected Course: " << selectedCourse << endl;
        file << "Course Level: " << courseLevel << endl;
        file << "Preferred Way of Learning: " << preferredWayOfLearning <<
endl;

        file << "Duration for Each Day: " << durationForEachDay << endl;
        file << "-----" << endl;
        file.close();
        cout << "Data saved to elearning_data.txt" << endl;
    } else {
        cout << "Unable to open file" << endl;
    }
}

};

int main() {
    userinfo();

    elearning obj1;
    obj1.selectCourse();
    int userChoice;
    cout << "Enter your choice: ";
    cin >> userChoice;
    cin.ignore();

    elearning obj2(userChoice);

    return 0;
}

```

Output:-

```

PS C:\Users\dwijd\Desktop\inClass_C++> cd "c:\Users\dwijd\Desktop\inClass_C++\" ; if ($?) { g++ pPRACTICAL_9.cpp -o pPRACTICAL_9 } ; if ($?) { .\pPRACTICAL_9 }
Enter your info here (Age, Name, Address, Email): 23 ender dwij dwijdvd

Concatenated string: 23 ender dwij dwijdvd
Length of concatenated string: 21
Enter the number of characters to display from the end: 2
Last 2 characters: vd
First character: 2
Last character: d
Enter the starting index and length of the substring to extract: 1
1
Extracted substring: 3
Enter the search term: 2
Match found!
>>>Course Selection<<<
<1> Business
<2> Computer course
<3> Data science
<4> ICT
Enter your choice: 1

You chose Business

>>>Enter choice for Computer course:<<<
<1> Course Level
<2> Preferred Way of Learning
<3> Duration for Each Day
<4> Exit
2
Preferred Way of Learning:
<1> Guided project
<2> Personal practical
<3> Notes and concept learning
2
Personal practical

>>>Enter choice for Computer course:<<<
<1> Course Level
<2> Preferred Way of Learning
<3> Duration for Each Day
<4> Exit
3
Duration for Each Day:
<1> 2 hours
<2> 4 hours
<3> 6 hours
1
2 hours

>>>Enter choice for Computer course:<<<
<1> Course Level
<2> Preferred Way of Learning
<3> Duration for Each Day
<4> Exit
4
Exiting...
Total duration to complete the course: 0 weeks
Data saved to elearning_data.txt
PS C:\Users\dwijd\Desktop\inClass_C++> █

```

Recorded data in text file:-

```

elearning_data.txt
1 Selected Course: Business
2 Course Level:
3 Preferred Way of Learning: Personal practical
4 Duration for Each Day: 2 hours
5 -----
6

```

```

1 #include <iostream>
2 #include <iomanip>
3 #include <string>
4
5 using namespace std;
6
7 void main() {
8     string name, name2, address, email;
9
10    cout << "Enter your 1st Name (Last, First, Address, Email): ";
11    cin >> name >> name2 >> address >> email;
12    cout << endl;
13
14    // Display name, address, email
15    string name2 = name + " " + name2 + " " + address + " " + email;
16    cout << "Name: " << name2 << endl;
17
18    // Display length of the concatenated string
19    cout << "Length of concatenated string: " << (name2.length()) << endl;
20
21    // Display last characters based on user choice
22    int lastChars;
23    cout << "Enter the number of characters to display from the end: ";
24    cin >> lastChars;
25    cout << "Last " << lastChars << " characters: " << (name2.substr(name2.length() - lastChars) << endl;
26
27    // Find and display first and last characters
28    cout << "First character: " << (name2[0]) << endl;
29    cout << "Last character: " << (name2[name2.length() - 1]) << endl;
30
31    // Extract substring based on user input
32    int start, end, length;
33    cout << "Enter the start, end, and length of the substring to extract: ";
34    cin >> start >> end >> length;
35    cout << "Extracted substring: " << (name2.substr(start, length)) << endl;
36
37    // Search within
38    string searchWord;
39    cout << "Enter the search word: ";
40    cin >> searchWord;
41    int foundPosition = name2.find(searchWord);
42    if (foundPosition != string::npos) {
43        cout << "Match found at: " << foundPosition << endl;
44    } else {
45        cout << "No match found." << endl;
46    }
47
48    // Class structure
49    struct {
50        int id;
51        string name;
52        string address;
53        string email;
54        string phone;
55    };
56
57    // Create array of struct
58    struct person {
59        int id;
60        string name;
61        string address;
62        string email;
63        string phone;
64    };
65
66    // Create array of struct
67    person p[10];
68
69    // Create array of struct
70    p[0].id = 1;
71    p[0].name = "John Doe";
72    p[0].address = "123 Main St";
73    p[0].email = "john.doe@example.com";
74    p[0].phone = "123-456-7890";
75
76    // Create array of struct
77    p[1].id = 2;
78    p[1].name = "Jane Smith";
79    p[1].address = "456 Elm St";
80    p[1].email = "jane.smith@example.com";
81    p[1].phone = "987-654-3210";
82
83    // Create array of struct
84    p[2].id = 3;
85    p[2].name = "Bob Johnson";
86    p[2].address = "789 Oak St";
87    p[2].email = "bob.johnson@example.com";
88    p[2].phone = "555-123-4567";
89
90    // Create array of struct
91    p[3].id = 4;
92    p[3].name = "Alice Brown";
93    p[3].address = "101 Pine St";
94    p[3].email = "alice.brown@example.com";
95    p[3].phone = "222-333-4444";
96
97    // Create array of struct
98    p[4].id = 5;
99    p[4].name = "Charlie Davis";
100    p[4].address = "202 Cedar St";
101    p[4].email = "charlie.davis@example.com";
102    p[4].phone = "777-888-9999";
103
104    // Create array of struct
105    p[5].id = 6;
106    p[5].name = "Diana Evans";
107    p[5].address = "303 Birch St";
108    p[5].email = "diana.evans@example.com";
109    p[5].phone = "666-777-8888";
110
111    // Create array of struct
112    p[6].id = 7;
113    p[6].name = "Frank Green";
114    p[6].address = "404 Spruce St";
115    p[6].email = "frank.green@example.com";
116    p[6].phone = "555-666-7777";
117
118    // Create array of struct
119    p[7].id = 8;
120    p[7].name = "Grace Hill";
121    p[7].address = "505 Ash St";
122    p[7].email = "grace.hill@example.com";
123    p[7].phone = "444-555-6666";
124
125    // Create array of struct
126    p[8].id = 9;
127    p[8].name = "Henry King";
128    p[8].address = "606 Elm St";
129    p[8].email = "henry.king@example.com";
130    p[8].phone = "333-444-5555";
131
132    // Create array of struct
133    p[9].id = 10;
134    p[9].name = "Ivy Lee";
135    p[9].address = "707 Oak St";
136    p[9].email = "ivy.lee@example.com";
137    p[9].phone = "222-333-4444";
138
139    // Create array of struct
140    p[10].id = 11;
141    p[10].name = "Jack Miller";
142    p[10].address = "808 Pine St";
143    p[10].email = "jack.miller@example.com";
144    p[10].phone = "111-222-3333";
145
146    // Create array of struct
147    p[11].id = 12;
148    p[11].name = "Karen Wilson";
149    p[11].address = "909 Cedar St";
150    p[11].email = "karen.wilson@example.com";
151    p[11].phone = "000-111-2222";
152
153    // Create array of struct
154    p[12].id = 13;
155    p[12].name = "Leo White";
156    p[12].address = "1010 Birch St";
157    p[12].email = "leo.white@example.com";
158    p[12].phone = "999-000-1111";
159
160    // Create array of struct
161    p[13].id = 14;
162    p[13].name = "Mia Young";
163    p[13].address = "1111 Spruce St";
164    p[13].email = "mia.young@example.com";
165    p[13].phone = "888-999-0000";
166
167    // Create array of struct
168    p[14].id = 15;
169    p[14].name = "Noah Adams";
170    p[14].address = "1212 Ash St";
171    p[14].email = "noah.adams@example.com";
172    p[14].phone = "777-888-9999";
173
174    // Create array of struct
175    p[15].id = 16;
176    p[15].name = "Olivia Baker";
177    p[15].address = "1313 Elm St";
178    p[15].email = "olivia.baker@example.com";
179    p[15].phone = "666-777-8888";
180
181    // Create array of struct
182    p[16].id = 17;
183    p[16].name = "Peter Clark";
184    p[16].address = "1414 Oak St";
185    p[16].email = "peter.clark@example.com";
186    p[16].phone = "555-666-7777";
187
188    // Create array of struct
189    p[17].id = 18;
190    p[17].name = "Quinn Evans";
191    p[17].address = "1515 Pine St";
192    p[17].email = "quinn.evans@example.com";
193    p[17].phone = "444-555-6666";
194
195    // Create array of struct
196    p[18].id = 19;
197    p[18].name = "Samuel Foster";
198    p[18].address = "1616 Cedar St";
199    p[18].email = "samuel.foster@example.com";
200    p[18].phone = "333-444-5555";
201
202    // Create array of struct
203    p[19].id = 20;
204    p[19].name = "Tina Green";
205    p[19].address = "1717 Birch St";
206    p[19].email = "tina.green@example.com";
207    p[19].phone = "222-333-4444";
208
209    // Create array of struct
210    p[20].id = 21;
211    p[20].name = "Uma Hall";
212    p[20].address = "1818 Spruce St";
213    p[20].email = "uma.hall@example.com";
214    p[20].phone = "111-222-3333";
215
216    // Create array of struct
217    p[21].id = 22;
218    p[21].name = "Victor King";
219    p[21].address = "1919 Ash St";
220    p[21].email = "victor.king@example.com";
221    p[21].phone = "000-111-2222";
222
223    // Create array of struct
224    p[22].id = 23;
225    p[22].name = "Wendy Lee";
226    p[22].address = "2020 Elm St";
227    p[22].email = "wendy.lee@example.com";
228    p[22].phone = "999-000-1111";
229
230    // Create array of struct
231    p[23].id = 24;
232    p[23].name = "Xavier Miller";
233    p[23].address = "2121 Oak St";
234    p[23].email = "xavier.miller@example.com";
235    p[23].phone = "888-999-0000";
236
237    // Create array of struct
238    p[24].id = 25;
239    p[24].name = "Yara Wilson";
240    p[24].address = "2222 Pine St";
241    p[24].email = "yara.wilson@example.com";
242    p[24].phone = "777-888-9999";
243
244    // Create array of struct
245    p[25].id = 26;
246    p[25].name = "Zoe Adams";
247    p[25].address = "2323 Cedar St";
248    p[25].email = "zoe.adams@example.com";
249    p[25].phone = "666-777-8888";
250
251    // Create array of struct
252    p[26].id = 27;
253    p[26].name = "Adam Baker";
254    p[26].address = "2424 Birch St";
255    p[26].email = "adam.baker@example.com";
256    p[26].phone = "555-666-7777";
257
258    // Create array of struct
259    p[27].id = 28;
260    p[27].name = "Bella Clark";
261    p[27].address = "2525 Spruce St";
262    p[27].email = "bella.clark@example.com";
263    p[27].phone = "444-555-6666";
264
265    // Create array of struct
266    p[28].id = 29;
267    p[28].name = "Caleb Evans";
268    p[28].address = "2626 Ash St";
269    p[28].email = "caleb.evans@example.com";
270    p[28].phone = "333-444-5555";
271
272    // Create array of struct
273    p[29].id = 30;
274    p[29].name = "Diana Foster";
275    p[29].address = "2727 Elm St";
276    p[29].email = "diana.foster@example.com";
277    p[29].phone = "222-333-4444";
278
279    // Create array of struct
280    p[30].id = 31;
281    p[30].name = "Ethan Green";
282    p[30].address = "2828 Oak St";
283    p[30].email = "ethan.green@example.com";
284    p[30].phone = "111-222-3333";
285
286    // Create array of struct
287    p[31].id = 32;
288    p[31].name = "Fiona Hall";
289    p[31].address = "2929 Pine St";
290    p[31].email = "fiona.hall@example.com";
291    p[31].phone = "000-111-2222";
292
293    // Create array of struct
294    p[32].id = 33;
295    p[32].name = "Gavin King";
296    p[32].address = "3030 Cedar St";
297    p[32].email = "gavin.king@example.com";
298    p[32].phone = "999-000-1111";
299
300    // Create array of struct
301    p[33].id = 34;
302    p[33].name = "Hannah Lee";
303    p[33].address = "3131 Birch St";
304    p[33].email = "hannah.lee@example.com";
305    p[33].phone = "888-999-0000";
306
307    // Create array of struct
308    p[34].id = 35;
309    p[34].name = "Ian Miller";
310    p[34].address = "3232 Spruce St";
311    p[34].email = "ian.miller@example.com";
312    p[34].phone = "777-888-9999";
313
314    // Create array of struct
315    p[35].id = 36;
316    p[35].name = "Julia Wilson";
317    p[35].address = "3333 Ash St";
318    p[35].email = "julia.wilson@example.com";
319    p[35].phone = "666-777-8888";
320
321    // Create array of struct
322    p[36].id = 37;
323    p[36].name = "Kevin Adams";
324    p[36].address = "3434 Elm St";
325    p[36].email = "kevin.adams@example.com";
326    p[36].phone = "555-666-7777";
327
328    // Create array of struct
329    p[37].id = 38;
330    p[37].name = "Liam Baker";
331    p[37].address = "3535 Oak St";
332    p[37].email = "liam.baker@example.com";
333    p[37].phone = "444-555-6666";
334
335    // Create array of struct
336    p[38].id = 39;
337    p[38].name = "Mia Clark";
338    p[38].address = "3636 Pine St";
339    p[38].email = "mia.clark@example.com";
340    p[38].phone = "333-444-5555";
341
342    // Create array of struct
343    p[39].id = 40;
344    p[39].name = "Nora Evans";
345    p[39].address = "3737 Cedar St";
346    p[39].email = "nora.evans@example.com";
347    p[39].phone = "222-333-4444";
348
349    // Create array of struct
350    p[40].id = 41;
351    p[40].name = "Oscar Foster";
352    p[40].address = "3838 Birch St";
353    p[40].email = "oscar.foster@example.com";
354    p[40].phone = "111-222-3333";
355
356    // Create array of struct
357    p[41].id = 42;
358    p[41].name = "Pamela Green";
359    p[41].address = "3939 Spruce St";
360    p[41].email = "pamela.green@example.com";
361    p[41].phone = "000-111-2222";
362
363    // Create array of struct
364    p[42].id = 43;
365    p[42].name = "Quinn Hall";
366    p[42].address = "4040 Ash St";
367    p[42].email = "quinn.hall@example.com";
368    p[42].phone = "999-000-1111";
369
370    // Create array of struct
371    p[43].id = 44;
372    p[43].name = "Rory King";
373    p[43].address = "4141 Elm St";
374    p[43].email = "rory.king@example.com";
375    p[43].phone = "888-999-0000";
376
377    // Create array of struct
378    p[44].id = 45;
379    p[44].name = "Sara Lee";
380    p[44].address = "4242 Oak St";
381    p[44].email = "sara.lee@example.com";
382    p[44].phone = "777-888-9999";
383
384    // Create array of struct
385    p[45].id = 46;
386    p[45].name = "Tina Miller";
387    p[45].address = "4343 Pine St";
388    p[45].email = "tina.miller@example.com";
389    p[45].phone = "666-777-8888";
390
391    // Create array of struct
392    p[46].id = 47;
393    p[46].name = "Uma Wilson";
394    p[46].address = "4444 Cedar St";
395    p[46].email = "uma.wilson@example.com";
396    p[46].phone = "555-666-7777";
397
398    // Create array of struct
399    p[47].id = 48;
400    p[47].name = "Victor Adams";
401    p[47].address = "4545 Birch St";
402    p[47].email = "victor.adams@example.com";
403    p[47].phone = "444-555-6666";
404
405    // Create array of struct
406    p[48].id = 49;
407    p[48].name = "Wendy Baker";
408    p[48].address = "4646 Spruce St";
409    p[48].email = "wendy.baker@example.com";
410    p[48].phone = "333-444-5555";
411
412    // Create array of struct
413    p[49].id = 50;
414    p[49].name = "Xavier Clark";
415    p[49].address = "4747 Ash St";
416    p[49].email = "xavier.clark@example.com";
417    p[49].phone = "222-333-4444";
418
419    // Create array of struct
420    p[50].id = 51;
421    p[50].name = "Yara Evans";
422    p[50].address = "4848 Elm St";
423    p[50].email = "yara.evans@example.com";
424    p[50].phone = "111-222-3333";
425
426    // Create array of struct
427    p[51].id = 52;
428    p[51].name = "Zoe Foster";
429    p[51].address = "4949 Oak St";
430    p[51].email = "zoe.foster@example.com";
431    p[51].phone = "000-111-2222";
432
433    // Create array of struct
434    p[52].id = 53;
435    p[52].name = "Adam Green";
436    p[52].address = "5050 Pine St";
437    p[52].email = "adam.green@example.com";
438    p[52].phone = "999-000-1111";
439
440    // Create array of struct
441    p[53].id = 54;
442    p[53].name = "Bella Hall";
443    p[53].address = "5151 Cedar St";
444    p[53].email = "bella.hall@example.com";
445    p[53].phone = "888-999-0000";
446
447    // Create array of struct
448    p[54].id = 55;
449    p[54].name = "Caleb King";
450    p[54].address = "5252 Birch St";
451    p[54].email = "caleb.king@example.com";
452    p[54].phone = "777-888-9999";
453
454    // Create array of struct
455    p[55].id = 56;
456    p[55].name = "Diana Lee";
457    p[55].address = "5353 Spruce St";
458    p[55].email = "diana.lee@example.com";
459    p[55].phone = "666-777-8888";
460
461    // Create array of struct
462    p[56].id = 57;
463    p[56].name = "Ethan Miller";
464    p[56].address = "5454 Ash St";
465    p[56].email = "ethan.miller@example.com";
466    p[56].phone = "555-666-7777";
467
468    // Create array of struct
469    p[57].id = 58;
470    p[57].name = "Fiona Wilson";
471    p[57].address = "5555 Elm St";
472    p[57].email = "fiona.wilson@example.com";
473    p[57].phone = "444-555-6666";
474
475    // Create array of struct
476    p[58].id = 59;
477    p[58].name = "Gavin Adams";
478    p[58].address = "5656 Oak St";
479    p[58].email = "gavin.adams@example.com";
480    p[58].phone = "333-444-5555";
481
482    // Create array of struct
483    p[59].id = 60;
484    p[59].name = "Hannah Baker";
485    p[59].address = "5757 Pine St";
486    p[59].email = "hannah.baker@example.com";
487    p[59].phone = "222-333-4444";
488
489    // Create array of struct
490    p[60].id = 61;
491    p[60].name = "Ian Clark";
492    p[60].address = "5858 Cedar St";
493    p[60].email = "ian.clark@example.com";
494    p[60].phone = "111-222-3333";
495
496    // Create array of struct
497    p[61].id = 62;
498    p[61].name = "Julia Evans";
499    p[61].address = "5959 Birch St";
500    p[61].email = "julia.evans@example.com";
501    p[61].phone = "000-111-2222";
502
503    // Create array of struct
504    p[62].id = 63;
505    p[62].name = "Kevin Foster";
506    p[62].address = "6060 Spruce St";
507    p[62].email = "kevin.foster@example.com";
508    p[62].phone = "999-000-1111";
509
510    // Create array of struct
511    p[63].id = 64;
512    p[63].name = "Liam Green";
513    p[63].address = "6161 Ash St";
514    p[63].email = "liam.green@example.com";
515    p[63].phone = "888-999-0000";
516
517    // Create array of struct
518    p[64].id = 65;
519    p[64].name = "Mia Hall";
520    p[64].address = "6262 Elm St";
521    p[64].email = "mia.hall@example.com";
522    p[64].phone = "777-888-9999";
523
524    // Create array of struct
525    p[65].id = 66;
526    p[65].name = "Nora King";
527    p[65].address = "6363 Oak St";
528    p[65].email = "nora.king@example.com";
529    p[65].phone = "666-777-8888";
530
531    // Create array of struct
532    p[66].id = 67;
533    p[66].name = "Oscar Lee";
534    p[66].address = "6464 Pine St";
535    p[66].email = "oscar.lee@example.com";
536    p[66].phone = "555-666-7777";
537
538    // Create array of struct
539    p[67].id = 68;
540    p[67].name = "Pamela Miller";
541    p[67].address = "6565 Cedar St";
542    p[67].email = "pamela.miller@example.com";
543    p[67].phone = "444-555-6666";
544
545    // Create array of struct
546    p[68].id = 69;
547    p[68].name = "Quinn Wilson";
548    p[68].address = "6666 Birch St";
549    p[68].email = "quinn.wilson@example.com";
550    p[68].phone = "333-444-5555";
551
552    // Create array of struct
553    p[69].id = 70;
554    p[69].name = "Rory Adams";
555    p[69].address = "6767 Spruce St";
556    p[69].email = "rory.adams@example.com";
557    p[69].phone = "222-333-4444";
558
559    // Create array of struct
560    p[70].id = 71;
561    p[70].name = "Sara Baker";
562    p[70].address = "6868 Ash St";
563    p[70].email = "sara.baker@example.com";
564    p[70].phone = "111-222-3333";
565
566    // Create array of struct
567    p[71].id = 72;
568    p[71].name = "Tina Clark";
569    p[71].address = "6969 Elm St";
570    p[71].email = "tina.clark@example.com";
571    p[71].phone = "000-111-2222";
572
573    // Create array of struct
574    p[72].id = 73;
575    p[72].name = "Uma Evans";
576    p[72].address = "7070 Oak St";
577    p[72].email = "uma.evans@example.com";
578    p[72].phone = "999-000-1111";
579
580    // Create array of struct
581    p[73].id = 74;
582    p[73].name = "Victor Foster";
583    p[73].address = "7171 Pine St";
584    p[73].email = "victor.foster@example.com";
585    p[73].phone = "888-999-0000";
586
587    // Create array of struct
588    p[74].id = 75;
589    p[74].name = "Wendy Green";
590    p[74].address = "7272 Cedar St";
591    p[74].email = "wendy.green@example.com";
592    p[74].phone = "777-888-9999";
593
594    // Create array of struct
595    p[75].id = 76;
596    p[75].name = "Xavier Hall";
597    p[75].address = "7373 Birch St";
598    p[75].email = "xavier.hall@example.com";
599    p[75].phone = "666-777-8888";
600
601    // Create array of struct
602    p[76].id = 77;
603    p[76].name = "Yara King";
604    p[76].address = "7474 Spruce St";
605    p[76].email = "yara.king@example.com";
606    p[76].phone = "555-666-7777";
607
608    // Create array of struct
609    p[77].id = 78;
610    p[77].name = "Zoe Lee";
611    p[77].address = "7575 Ash St";
612    p[77].email = "zoe.lee@example.com";
613    p[77].phone = "444-555-6666";
614
615    // Create array of struct
616    p[78].id = 79;
617    p[78].name = "Adam Miller";
618    p[78].address = "7676 Elm St";
619    p[78].email = "adam.miller@example.com";
620    p[78].phone = "333-444-5555";
621
622    // Create array of struct
623    p[79].id = 80;
624    p[79].name = "Bella Wilson";
625    p[79].address = "7777 Oak St";
626    p[79].email = "bella.wilson@example.com";
627    p[79].phone = "222-333-4444";
628
629    // Create array of struct
630    p[80].id = 81;
631    p[80].name = "Caleb Adams";
632    p[80].address = "7878 Pine St";
633    p[80].email = "caleb.adams@example.com";
634    p[80].phone = "111-222-3333";
635
636    // Create array of struct
637    p[81].id = 82;
638    p[81].name = "Diana Baker";
639    p[81].address = "7979 Cedar St";
640    p[81].email = "diana.baker@example.com";
641    p[81].phone = "000-111-2222";
642
643    // Create array of struct
644    p[82].id = 83;
645    p[82].name = "Ethan Clark";
646    p[82].address = "8080 Birch St";
647    p[82].email = "ethan.clark@example.com";
648    p[82].phone = "999-000-1111";
649
650    // Create array of struct
651    p[83].id = 84;
652    p[83].name = "Fiona Evans";
653    p[83].address = "8181 Spruce St";
654    p[83].email = "fiona.evans@example.com";
655    p[83].phone = "888-999-0000";
656
657    // Create array of struct
658    p[84].id = 85;
659    p[84].name = "Gavin Foster";
660    p[84].address = "8282 Ash St";
661    p[84].email = "gavin.foster@example.com";
662    p[84].phone = "777-888-9999";
663
664    // Create array of struct
665    p[85].id = 86;
666    p[85].name = "Hannah Green";
667    p[85].address = "8383 Elm St";
668    p[85].email = "hannah.green@example.com";
669    p[85].phone = "666-777-8888";
670
671    // Create array of struct
672    p[86].id = 87;
673    p[86].name = "Ian Hall";
674    p[86].address = "8484 Oak St";
675    p[86].email = "ian.hall@example.com";
676    p[86].phone = "555-666-7777";
677
678    // Create array of struct
679    p[87].id = 88;
680    p[87].name = "Julia King";
681    p[87].address = "8585 Pine St";
682    p[87].email = "julia.king@example.com";
683    p[87].phone = "444-555-6666";
684
685    // Create array of struct
686    p[88].id = 89;
687    p[88].name = "Kevin Lee";
688    p[88].address = "8686 Cedar St";
689    p[88].email = "kevin.lee@example.com";
690    p[88].phone = "333-444-5555";
691
692    // Create array of struct
693    p[89].id = 90;
694    p[89].name = "Liam Miller";
695    p[89].address = "8787 Birch St";
696    p[89].email = "liam.miller@example.com";
697    p[89].phone = "222-333-4444";
698
699    // Create array of struct
700    p[90].id = 91;
701    p[90].name = "Mia Wilson";
702    p[90].address = "8888 Spruce St";
703    p[90].email = "mia.wilson@example.com";
704    p[90].phone = "111-222-3333";
705
706    // Create array of struct
707    p[91].id = 92;
708    p[91].name = "Nora Adams";
709    p[91].address = "8989 Ash St";
710    p[91].email = "nora.adams@example.com";
711    p[91].phone = "000-111-2222";
712
713    // Create array of struct
714    p[92].id = 93;
715    p[92].name = "Oscar Baker";
716    p[92].address = "9090 Elm St";
717    p[92].email = "oscar.baker@example.com";
718    p[92].phone = "999-000-1111";
719
720    // Create array of struct
721    p[93].id = 94;
722    p[93].name = "Pamela Clark";
723    p[93].address = "9191 Oak St";
724    p[93].email = "pamela.clark@example.com";
725    p[93].phone = "888-999-0000";
726
727    // Create array of struct
728    p[94].id = 95;
729    p[94].name = "Quinn Evans";
730    p[94].address = "9292 Pine St";
731    p[94].email = "quinn.evans@example.com";
732    p[94].phone = "777-888-9999";
733
734    // Create array of struct
735    p[95].id = 96;
736    p[95].name = "Rory Foster";
737    p[95].address = "9393 Cedar St";
738    p[95].email = "rory.foster@example.com";
739    p[95].phone = "666-777-8888";
740
741    // Create array of struct
742    p[96].id = 97;
743    p[96].name = "Sara Green";
744    p[96].address = "9494 Birch St";
745    p[96].email = "sara.green@example.com";
746    p[96].phone = "555-666-7777";
747
748    // Create array of struct
749    p[97].id = 98;
750    p[97].name = "Tina Hall";
751    p[97].address = "9595 Spruce St";
752    p[97].email = "tina.hall@example.com";
753    p[97].phone = "444-555-6666";
754
755    // Create array of struct
756    p[98].id = 99;
757    p[98].name = "Uma King";
758    p[98].address = "9696 Ash St";
759    p[98].email = "uma.king@example.com";
760    p[98].phone = "333-444-5555";
761
762    // Create array of struct
763    p[99].id = 100;
764    p[99].name = "Victor Lee";
765    p[99].address = "9797 Elm St";
766    p[99].email = "victor.lee@example.com";
767    p[99].phone = "222-333-4444";
768
769    // Create array of struct
770    p[100].id = 101;
771    p[100].name = "Wendy Miller";
772    p[100].address = "9898 Oak St";
773    p[100].email = "wendy.miller@example.com";
774    p[100].phone = "111-222-3333";
775
776    // Create array of struct
777    p[101].id = 102;
778    p[101].name = "Xavier Wilson";
779    p[101].address = "9999 Pine St";
780    p[101].email = "xavier.wilson@example.com";
781    p[101].phone = "000-111-2222";
782
783    // Create array of struct
784    p[102].id = 103;
785    p[102].name = "Yara Adams";
786    p[102].address = "10000 Cedar St";
787    p[102].email = "yara.adams@example.com";
788    p[102].phone = "999-000-
```