# 02. Memory Management – Homework Exercises

Write C++ code for solving the tasks on the following pages.

Code should compile under the C++03 or the C++11 standard.

Submit your solutions here: <https://judge.softuni.bg/Contests/1152/02-Memory-Management-Homework> (select “Compete” when prompted)

Any code files that are part of the task are provided under the folder **Skeleton**.

Please follow the exact instructions on uploading the solutions for each task.

# Task 5 – Serialize

NOTE: this task is the reverse of Task 5 – Memory from the lecture 01. Pointers and References.

You are given program that reads information about **companies** and writes it to the console.

Each company has:

* An **id** (an integer between 0 and 255)
* A **name** (a string containing a sequence of lowercase English letters a-z)
* **Employees** by their initials (a vector of pairs of characters, containing at most 255 employee initials)

The program reads the information in its string representation and calls a function named serializeToMemory. The function should parse the companies from the input and then write them to memory as a sequence of bytes in a dynamically allocated array (the format is detailed below). The function will be called with the following two parameters:

* A string containing lines, where each line is the string representation of a Company
* An integer which the program expects to be set with the number of bytes serialized to memory which contain the representation of the companies from the first parameter

The program expects the function **serializeToMemory** to return a pointer to the memory where the companies have been written (serialized).

The memory format of each company is the following:

* the first byte contains the **id** of the company (0-255)
* the **name** of the company starts from the second byte and ends with a null terminator (the value 0, or '\0'), i.e. the name of the company is placed in memory the same way a null-terminated C-String would be
* the next byte contains the number of employees the company has (0-255). Let’s call it numEmployees
* the following numEmployees \* 2 bytes contain pairs of initials of the employees, i.e. if the numEmployees byte is at address x, then the **first employee’s first initial** is at address x + 1, their **second initial** is at address x + 2, the **second employee’s first initial** is at address x + 3 and their **second** is at address x + 4 and so on.

Additionally, since there can be more than one Company:

* the **first byte** in the memory describing the companies contains an **integer representing the number of companies** serialized

For example, if we have the companies:

* id = 42, name = "uni", employees = { {'I', 'K'}, {'S', 'N'} } and   
  id = 13, name = "joro", employees = { {'G', 'G' } }

Their representation as a string read by the program and passed to **serializeToMemory** will be:

"42 uni (I.K.,S.N.)\n13 joro (G.G.)"

Their representation in memory, assuming the memory starts at byte address M**,** will be:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Offset from start** | **+0** | **+1** | **+2** | **+3** | **+4** | **+5** | **+6** | **+7** | **+8** | **+9** | **+10** | **+11** | **+12** | **+13** | **+14** |
| **Value** | **2** | **42** | **'u'** | **'n'** | **'i'** | **'\0'** | **2** | **'I'** | **'K'** | **'S'** | **'N'** | **13** | **'j'** | **'o'** | **'r'** |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Offset from start** | **+14** | **+15** | **+16** | **+17** | **+18** |
| **Value** | **'o'** | **'\0'** | **1** | **'G'** | **'G'** |

And their representation in the output for the task will be:

2 42 117 110 105 0 2 73 75 83 78 13 106 111 114 111 0 1 71 71

Your task is to create a file called Serialize.h containing the function **serializeToMemory**, implemented in such a way that the program compiles and works as described.

You should submit a single .zip file for this task, containing ONLY the **Serialize**.h file. The Judge system has a copy of the other files and will compile them along with your **Serialize**.h file in the same directory.

Hint: the Company class supports reading from a stream, so you don’t need to implement the parsing of the string yourself. The following code reads companies from the **string**, until there are no more companies to read:

std::istringstream companiesIn(companiesString);

Company company;

while (companiesIn >> company) { }

### Examples

|  |  |
| --- | --- |
| **Input** | **Output (*NOTE: single line*)** |
| 42 uni (I.K.,S.N.)  13 joro (G.G.)  end | 2 42 117 110 105 0 2 73 75 83 78 13 106 111 114 111 0 1 71 71 |
| 188 icyha (B.Q.,H.P.,F.S.)  58 uadel (S.A.,C.H.,L.T.)  end | 2 188 105 99 121 104 97 0 3 66 81 72 80 70 83 58 117 97 100 101 108 0 3 83 65 67 72 76 84 |
| 13 joro (G.G.)  end | 1 13 106 111 114 111 0 1 71 71 |