# Task 5 – Serialize

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

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

Each company has:

* **id** (int 0 - 255)
* **name** (string a-z)
* **Employees** by initials (vector<pair<char, char> > containing at most 255 employee initials)

program reads info in its string representation and calls function serializeToMemory. function should parse companies from input and then write them to memory as sequence of bytes in dynamically allocated array. Function will be called with 2 parameters:

* string containing lines, where each line is string representation of Company
* **int** which program expects to be set with num of bytes serialized to memory which contain representation of companies from 1st parameter

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

memory format of company:

* 1st byte - **id** (0-255)
* **name** of the company starts from 2nd byte and ends with (value 0, or '\0')
* next byte contains numEmployees (0-255)
* following numEmployees \* 2 bytes contain pairs of initials

Additionally, since there can be more Company:

* **1st byte** in memory describing companies contains **int representing num of companies** serialized

example:

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

and   
id = 13, name = "joro", employees = { {'G', 'G' } }

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

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

Their representation in 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'** |

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

std::istringstream companiesIn(companiesString);

Company company;

while (companiesIn >> company) { }

|  |  |
| --- | --- |
| **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 |