

Letterkenny I.T., BSc Games Development, Year 3, Algorithms and Data Structures for Games, Spring Term 2009. Lecturer: J.G. Campbell

Assignment 1, Deadline for paper copy Wed. 11th February 2008, 10.30am. Demonstrations must be completed by Wed. 11th Feb. 10.30am. Worth 20% of module marks (one third of CA).

Late assignments without documented extenuating circumstances **will lose 10% of marks per day or part of day late. Those seven days late will receive zero marks.** If you have extenuating circumstances, you should contact me as soon as possible; in particular, be careful about wasting time on completing an assignment only to hand it in well after the deadline; whether you have valid reasons or not I cannot give any marks for work handed in after outline answers have been handed out or discussed in class.

Computing Department Rules. Assignments must typed and neatly presented and spell-checked. Sloppily presented work will be severely penalised or returned for correction and resubmission.

Poor layout of programs will be penalised.

Expect to be questioned about the content of assignments. I want assignments to be handed to me in the practical class of Wed. 11th Feb., so that I can ask questions.

Please use only one side of the page. There is no need for ornate front sheets or plastic binding; simply include the heading above (including my name), and your own name. A staple at the top left corner is the preferred binding – I can staple them if you wish. There is no need for plastic or other covering. Please use the Computing Department cover sheet and signed declaration.

Software will be available in my public folder and on my website www.jgcampbell.com/adsgp/.

1. Code in `ch02array`; just to show that you have the environment set up and have downloaded the correct software, it is important, before you attempt anything more ambitious, that you get these running. *100% of marks for demonstration.*
 - Compile and execute `ArrayT1.cpp`.
 - Compile and execute `vectorT1.cpp`.
 - Compile and execute `NameT1.cpp`.
 - Compile and execute `vectorT2.cpp`.

[20 marks]

2. *50% of marks for demonstration.* Include the complete program and its output in your report. First copy `ArrayT1.cpp` to `ArrayT5.cpp` and delete most of what is already in it.
 - (a) Add code to `ArrayT5.cpp`, using `push_back`, to create an Array with the surnames of six politicians. Print the Array (`cout`); program fragment and output required for the assignment report.
 - (b) Now use `insert` to insert your own surname in the middle (after the third element) of that array. Print the Array; program fragment and output required for the assignment report.

[2 × (10 + 10) = 40 marks]

3. *50% of marks for demonstration.* Repeat 2. using `std::vector`. Copy Note, for output, you will need to use copy, see my notes. First copy `vectorT1.cpp` to `vectorT5.cpp` and delete most of what is already in it.
 - (a) See 2.(a)
 - (b) See 2.(b)
 - (c) Use `sort` to sort the vector; show the result.

[3 × (10 + 10) = 60 marks]

4. *50% of marks for demonstration.* Repeat 2. and 3. using `std::list`. Note, for output, you will need to use copy, see my notes.

[3 × (10 + 10) = 60 marks]

5. 50% of marks for demonstration. Repeat 2., 3., and 4. using our doubly linked list in `List.h` (`ch05list`).

[3 × (10 + 10) = 60 marks]

6. 50% of marks for demonstration. Repeat 2., 3., 4., 5. using `std::vector` and `Name` objects; add first names to the politicians and yourself.

[3 × (10 + 10) = 60 marks]

Remove operator `<` from `Name.h` and `Name.cpp`; comment them out. Comment on what happens — concentrate on item (c).

[10 marks]

7. As N , the number of elements, increases to a very large number, what can we say about the memory usage efficiency for the four cases: (i) plain array, e.g. `int a[N]` or `int *pa = new int[N]`; (ii) `std::vector` (assume it uses the same representation as `Array.h`) (iii) singly linked list, (iv) doubly linked list. See notes, end of Chapter 5. (v) If any of (ii), (iii) or (iv) are mosr wasteful of memory than plain arrays, make an argument for using them.

[50 marks]

8. Code from `ListT1.cpp`. Assuming a doubly linked list, draw diagrams showing the state of the list (showing the three data members of `Link`) after `//A` and after each of the four insertions at `//B` and after `//C`.

```
ListI c5;
for(uint i = 0; i < 4; ++i){
    c5.push_back(i + 3); // B
}

cout << "ListI::Iterator itr2 = c5.begin(), ++, ++ " << endl;
cout << "c5.insert(itr2, 5, 3) " << endl;
ListI::Iterator itr2 = c5.begin();
itr2++; itr2++;
c5.insert(itr2, 2, 3); // C
cout << c5;
```

[20 marks]

9. When inserting data values into containers such as `std::vector` and `std::list`, it is recommended to prefer `push_back`. Comment on this recommendation.

[20 marks]

10. (a) `std::vector` has no `push_front` operation because it is reckoned this operation is so inefficient (poor performance) that it would a crazy bit of programming to use it. Why? Explain in terms of *big-Oh*.
(b) *Reallocation*, e.g. in `Array::push_back` when we exceed the capacity is highly inefficient. Why? Explain in terms of *big-Oh*.

[2 × 10 = 20 marks]

11. Give an *explanation* of the *big-Oh* growth rate of the following program fragments. When I say *explanation*, I mean give the *big-Oh* and a full explanation of how you arrived at it.

(a) for (int j = 0; j < N; j++){
 statements with $O(1)$
 }

(b) for (int j = 0; j < N; j++){
 statements with $O(N)$
 }

(c) for (int i = 0; i < N; i++){
 for (int j = 0; j < N; j++){
 statements with $O(1)$
 }
 }

 for (int k = 0; k < N; k++){
 statements with $O(1)$
 }

(d) for (int i = 0; i < N; i++){
 for (int j = 0; j < i; j++){
 statements with $O(1)$
 }
 }

(e) for (int k = 0; k < N; k++){
 for (int i = 0; i < N; i++){
 for (int j = i; j < N; j++){
 statements with $O(1)$
 }
 }
 }

(f) int m = N;
 while(m > 1){
 statements with $O(1)$ // A
 m = m/2;
 }

Hint: Try it for $N = 32$, $N = 16$, $N = 8$ etc. Hint 2: have a look at *binary search*.

Note: in any of the examples above, you could easily place a counter at the core of the loop (e.g. at //A and show how the count of executions of //A varies with N ; you could work out the $O(f(N))$ function f from that data.

[6 × 10 = 60 marks]

12. (a) Read up about the *Big-Three*: *copy constructor*, *assignment operator*, and *destructor* in Chapter 2. Class `Name` does not seem to provide any of these. Discuss.

[20 marks]

Note: Any class whose objects are to be contained in a container such as `std::vector`, `std::list` must provide an assignment operator (`operator =`).

(b) For `Name`, write down the (explicit) declarations of *copy constructor*, *assignment operator*, and *destructor* for `Name` and give examples of code fragments where each would be called.

[10 marks]