**1.** Why do we need data structures with varying numbers of elements?

In some cases the number of elements can not be defined during compiling time. If this number depends on the input of user or some other value acquired during the runtime, the memory for value(s) should be allocated dynamically.

**2.** What four kinds of storage do we have for a typical program?

1. Code(Text) segment - read-only segment of memory which holds the code of the program.

2. BSS segment - has both read-only and writable parts. Holds static and global variables that haven’t been initialized yet or holds the value of 0.

3. Data segment - similiar to BSS segment has both read-only and writable parts. Contains static and global variables that have been already initialized.

4. Heap (Free store) - dynamically allocated memory. Grows towards the Stack segment. When the heap pointer meets stack pointer, the free memory is exhausted.

5. Stack - holds the information about function being executed. Also it stores the local variables, arguments and return address of the function.

**3.** What is the free store? What other name is commonly used for it? What operators support it?

Free store (Heap) is a data structure used to organise dynamically-allocated segment of memory. To allocate a memory for an object, **new** operator should be used in C++

**4.** What is a dereference operator and why do we need one?

Derefernce operator (\*) used to get a value of an object, pointed to by a pointer. We need it to interact with an object that was dynamically allocated.

**5.** What is an address? How are memory addresses manipulated in C++?

Address is a value that allows us to find the object in the memory. It can be manipulated with pointers.

**6.** What information about a pointed-to object does a pointer have? What useful information does it lack?

It has a type of object it points to but lacks the number of elements in an array.

**7.** What can a pointer point to?

To the beginning of the object in the memory.

**8.** What is a leak?

Losing control over the dynamically allocated memory, which can’t be used anymore by the computer to allocate it for another data.

**9.** What is a resource?

Something that program takes and must give back (release). E.g. dynamically allocated memory.

**10.** How can we initialize a pointer?

By using \* at the end of the type and assigning the pointer to dynamically allocated object. int\* p = new int(1);

**11.** What is a null pointer? When do we need to use one?

Null pointer doesn’t point to a valid object. To define it **nullptr** keyword should be used. We use it, when the pointer sometimes point to an object and sometimes not. It can be used to check whether pointer points to some object or not.

**12.** When do we need a pointer (instead of a reference or a named object)?

We prefer to use pointer in cases, where the empty object, which can be represented by **nullptr**, is valid.

**13.** What is a destructor? When do we want one?

Destructor is used to deallocate the space, that was once occupied by the dynamically-allocated objects of the class. We should use one when we have dynamically-allocated objects within the class.

**14.** When do we want a virtual destructor?

If there is a virtual function within a base classes and it is likely that dynamically-allocated object of derived class will be manipulated through the pointer to the base class, virtual destructor should be defined.

**15.** How are destructors for members called?

Destructors of class members are called implicitly through the destructor of base class.

**16.** What is a cast? When do we need to use one?

Cast is an explicit change of the object’s type. static\_cast should be used to explicitly convert the pointer of type **void\*** to the type, that can be dereferenced. This situations can occur when we need to interact with the code written in another language or old code, that was written without awareness of type safety.

**17.** How do we access a member of a class through a pointer?

With the help of ->(arrow) operator

**18.** What is a doubly-linked list?

It is a list, each member of which has a connection with the next and previous element of this list.

**19.** What is this and when do we need to use it?

**this** operator points to the object within a call of member function. We use it when we need to use a pointer to the current object (assign a pointer to some member of the class or return it for use in another part of program)