```
#include <iostream>
#include <memory>
#include <vector>
#include <set>
#include <algorithm>
#include <string>



int main()
{

/*************************************************************/
//Задание 1.
	/*
	Реализуйте шаблон функции для печати любых последовательностей (vector, list, deque, set и встроенного массива), которые могут содержать:
	•	как объекты любого типа,
	•	так и указатели на объекты любого типа (указатели распечатывать неинтересно => в этом случае следует получать значение по адресу)
	Подсказки: if constexpr
	*/
	{

	}

/*************************************************************/
//Задание 2.
	/* Реализуйте шаблон функции сложения двух значений.
	Если первое слагаемое является вектором, то все элементы вектора нужно увеличить на значение второго параметра. При этом элементы вектора и второй параметр должны быть одного и того же типа.
	Подсказки: if constexpr, is_same
	*/
	{

	}


	/*************************************************************/
//Задание 3.
	/*	Реализуйте шаблон функции вывода на печать значений элементов любого адаптера (stack, queue, priority_queue)
	Подсказки: if constexpr, is_same
	Предусмотрите вывод значений, если в адаптере хранятся указатели.
	*/
	{

	}

/*************************************************************/
//Задание 4.
	{
		//Дан массив элементов типа string
		std::string strings[] = {"abc", "123", "qwerty", "#$%"};
		//До завершения фрагмента строки должны существовать в единственном экземпляре.
		//Требуется обеспечить манипулирование строками а) без копирования и б) без изменения порядка
```

```
            //элементов в массиве!

            //В std::set "складываем" по алфавиту обертки для строк,
которые содержат только буквы


            __asm nop

      /**********************************************************************
**********************/

            //В std::vector "складываем" обертки для строк, которые
содержат только цифры
            //Выводим на экран
            //Находим сумму

            //std::vector<std::shared_ptr < std::string>>


      /**********************************************************************
**********************/
            //сюда "складываем" обертки для строк, которые не содержат ни
символов букв, ни символов цифр
            //и просто выводим


      }


/**************************************************************************
*****************/
//Задание 5.
      {
            //Дано:
            std::string ar[] = {"my","Hello", "World"};
            std::vector < std::shared_ptr<std::string>> v =
{std::make_shared<std::string>("good"),
std::make_shared<std::string>("bye")};



            //a) Требуется добавить в вектор обертки для элементов
массива, НЕ копируя элементы массива!
            //б) Отсортировать вектор по алфавиту и вывести на экран
            //в) Обеспечить корректное освобождение памяти


            __asm nop
      }
      /*****************************************************************/
            //Задание 6. shared_ptr и weak_ptr
            //Создаем генеалогическое дерево посредством класса human. В
классе хранятся:
            //имя - string
            //возможно признак: жив или уже нет...
            //родители - shared_ptr (родители не всегда известны...)
            //дети - контейнер из weak_ptr (чтобы избежать циклических
зависимостей)

            //Методы класса human:
            //конструктор - для инициализации имени и признака
```

```
            //конструктор копирования, оператор присваивания, move ???
            //статический метод child() -
            //                      должен создать создать и вернуть
обертку для родившегося человека
            //                      + сформировать все связи ребенка с
родителями и наоборот

            //Ввести возможность распечатать генеалогическое дерево для
указанного индивидума

      {
            //История должна с кого-то начинаться => "Жили-были дед да
баба, например, Адам и Ева"
            //(то есть на самом деле два деда и две бабы):


            //std::shared_ptr<human> grandM1(new human("Eva"));
            //...

            //у них появились дети - child():


            //а у детей в свою очередь свои дети:


            //...
            __asm nop
      }



//*************************************************************/
//Задание 7.

      /*Пользовательский deduction guide – для вывода типов параметров
шаблона
      Задан шаблон класса, который инкапсулирует внедренный ограниченный
массив известной
      размерности с элементами любого типа. */
/*
template<typename T, size_t size> class MyArray
      {
            T ar[size]; //как обеспечить инициализацию элементов базового
типа по умолчанию нулем?
            …
      };

*/
/*
//Требуется обеспечить работоспособность приведенных примеров
использования.
      {
            MyArray<int, 5> ar1;//MyArray<int,5>
            MyArray ar2{"ABC"}; //MyArray<char,4>
            int ar[] = { 1,2,3 };
            MyArray ar3{ ar };

      }
*/

      __asm nop
```

}