Задача 1. Задачата да се реши на езика C++.

1) Нека е дефиниран масивът
`int arr[] = { 1, 2, 3 };`
Срещу всеки от изразите да се посочи каква ще бъде неговата оценка.

```
arr[1] == *(arr+2) _____________________
arr == &arr[0]     _____________________
(arr+1) == &arr[1] _____________________
*arr == arr[0]     _____________________
```

2) Нека е дадена следната дефиниция:

```
void mystery(const char* str)
{
  while (*str && *(str+1)) {
    std::cout << *str;
    str += 2;
  }
}
```

Да се посочи какво ще изведе на екрана обръщението:

```
mystery("abcdef");
```

_____

3) Нека са дадени следните дефиниции:

```
char s1[] = "Hello";
char s2[] = "world!";
char result[80];
```

Да се довърши програмният фрагмент, така че след изпълнението му в `result` да се съхрани коректното представяне на низа `"Hello world!"`. На празните места трябва да се попълнят имената на подходящи стандартни функции за работа с низове.

```
_____________________(result, s1);
_____________________(result, " ");
_____________________(result, s2);
```

4) Да се довърши кодът на рекурсивните функции, така че `f` да проверява дали символният низ, сочен от `word`, се съдържа като подниз в `text`. За определеност считаме, че празният низ се съдържа във всеки друг.

```
bool g(const char* text, const char* word)
{
  if (!*word) return true;
  if (!*text) return _______________;
  if (*word != *text) return false;
  return g(_______________, _______________);
}

bool f(const char* text, const char* word)
{
  if (!*word) return _______________;
  if (!*text) return false;
  return g(_______________, _______________) ||
         f(_______________, _______________);
}
```

5) Да се посочи какво ще изведе на екрана даденият по-долу фрагмент:

```
char arr[3][3] = { 'a', 'b', 'c',
                   'd', 'e', 'f',
                   'g', 'h', 'i' };
for (int i = 0; i < 3; ++i)
  std::cout << arr[2-i][i];
```

_____

6) Да се посочи какво ще изведе на екрана даденият по-долу фрагмент:

```
double var = 5 / 2;
std::cout << var;
```

_____

Задача 2. Задачата да се реши на езика C++.

1) Освен конструктора по подразбиране (default constructor), кои други функции влизат в "голямата четворка" (функциите от т.нар. "rule-of-3")? Да се попълнят имената им в полетата долу:

_____
_____
_____

2) Нека е дадена дефиницията:

```
class foo {
public:
  virtual void f() {};
  void g() {};
};
```

Срещу всеки от редовете, които извикват `f` или `g`, да се запише "статично" или "динамично" според вида свързване, който ще се използва за тях.

```
foo obj;
foo& ref = obj;
obj.f();  ____________________
obj.g();  ____________________
ref.f();  ____________________
ref.g();  ____________________
```

3) Нека са дадени следните дефиниции:

```
class base {
public:  int a;
private: int b;
};
class derived : protected base { };
```

Да се посочи каква ще бъде видимостта на променливите `a` и `b` в класа `derived` – `public`, `protected` или `private`.

- Видимост на `a`: _____
- Видимост на `b`: _____

4) Нека класът `X` е абстрактен. Срещу всяко от твърденията да се посочи "да" или "не" според това дали е вярно:

- Могат да се създават обекти от тип `X`: ____
- Могат да се създават референции (reference) към обекти от тип `X`: ____

5) Нека е дадена следната дефиниция:

```
struct s {
public:
  static int var;
  s() { var = 5; }
};
int s::var = 0;
```

Да се посочи какво ще изведе следният фрагмент:

```
std::cout << '(' << s::var << ')';
s obj1;
obj1.var = 10;
s obj2;
std::cout << '-' << s::var << '-';
```

_____

6) Да се допълни дефиницията на класа `test`, така че функцията `f` да бъде чиста виртуална (pure-virtual) и класът да може коректно да се използва като основа на полиморфна йерархия.

```
class test {
public:
  _____________ void f() _____________;
  ____________________________________
};
```

7) Да се допълни дефиницията на шаблона `Array`, така че функцията `test` да се компилира коректно и да извежда на стандартния изход `55`.

```
_____________ <_____________>
class Array {
  static const size_t size = 10;
  T data[size];
public:
  _____________ at(size_t index) {
    if (index _____________________)
      throw std::out_of_range("error");
    return data[index];
  }
};

void test() {
  Array<int> a;
  a.at(0) = 5;
  std::cout << a.at(0);
  Array<Array<int>> b;
  b.at(0) = a;
  std::cout << b.at(0).at(0);
}
```