Задача 1 (20 точки)

Кодът не се компилира, защото `print_just_number` приема неконстанта референция. (10 точки) Трябва тази референция да бъде константна, защото референцията, която приемаме от `pretty_print` е константа. Референциите са константи, защото никъде не променяме стойностите им (10 точки).

Задача 2 (20 точки)

Понеже инстанцията `b` се унищожава като приключи нейния scope, във вектора остава указател, който сочи към вече изтрит обект.

Задача 3 (80 точки)

```
#include <iostream>
#include <cstring>
#include <fstream>
class Person{
public:
    Person();
    Person(const char* new_first_name, const char* new_last_name);
    Person(const Person& from);
    Person& operator=(const Person& from);
    ~Person();

    void set_first_name(const char* new_first_name);
    void set_last_name(const char* new_last_name);
    void write_to_file(const char* path) const;

    char* get_first_name() const;
    char* get_last_name() const;
private:

    static char* allocate_copy_string(const char* source);
    static void deallocate_string(char* source);

    char* first_name;
    char* last_name;
};

char* Person::allocate_copy_string(const char* source){
    int length = strlen(source);

    char* result = new char[length + 1];

    strcpy(result, source);

    result[length] = '\0';
    return result;
}
void Person::deallocate_string(char* source){
    delete[] source;
}

Person::Person(){
    this->first_name = Person::allocate_copy_string("\0");
    this->last_name = Person::allocate_copy_string("\0");
}

Person::Person(const char* new_first_name, const char* new_last_name){
    this->first_name = Person::allocate_copy_string(new_first_name);
    this->last_name = Person::allocate_copy_string(new_last_name);
}
Person::Person(const Person& from){
    this->first_name = Person::allocate_copy_string(from.first_name);
    this->last_name = Person::allocate_copy_string(from.last_name);
}

Person& Person::operator=(const Person& from){
    if(this != &from)
    {
        Person::deallocate_string(this->first_name);
        Person::deallocate_string(this->last_name);

        this->first_name = Person::allocate_copy_string(from.first_name);
        this->last_name = Person::allocate_copy_string(from.last_name);
    }
    return *this;
}

Person::~Person(){
    Person::deallocate_string(this->first_name);
    Person::deallocate_string(this->last_name);
}

void Person::set_first_name(const char* new_first_name){
    Person::deallocate_string(this->first_name);
    this->first_name = Person::allocate_copy_string(new_first_name);
}

void Person::set_last_name(const char* new_last_name){
    Person::deallocate_string(this->last_name);
    this->last_name = Person::allocate_copy_string(new_last_name);
}

void Person::write_to_file(const char* path) const{
    std::ofstream result(path, std::ios::in);

    if(result.is_open()){
        result << this;
    }

    result.close();
}
char* Person::get_first_name() const{
    return this->first_name;
}
char* Person::get_last_name() const{
    return this->last_name;
}
```

```
std::ostream& operator<<(std::ostream& out, const Person& to_print){
    out << to_print.get_first_name() << " " << to_print.get_last_name();

    return out;
}

std::istream& operator>>(std::istream& in, Person& to_write){
    char first_name[100];
    in>>first_name;

    char last_name[100];
    in>>last_name;

    to_write = Person(first_name, last_name);

    return in;
}
int main(){
    std::cout << "Hello world !";
}
```

- Голяма четворка (всяко по 8 точки) - 32 точки
- Get - всеки по 2 точки
- Set - всеки по 2 точки
- Оператор<< - 10 точки
- Оператор>> - 10 точки
- Писане във файл - 10 точки
- const, const& - 5 точки
- Изнасяне на логика за работа с динамична памет - 5 точки

Задача 4 (80 + 10 точки)

```
#include <iostream>
#include <vector>
#include <string>
class MultimediaFile{
public:
    MultimediaFile() = default;
    MultimediaFile(const std::string& init_name, const std::string&
init_extension, const unsigned& init_size);
    virtual ~MultimediaFile() = default;

    void set_name(const std::string& new_name);
    void set_extension(const std::string& new_extension);
    void set_size(const unsigned& new_size);

    virtual void play();

    std::string get_name() const;
    std::string get_extension() const;
    unsigned get_size() const;
private:
    std::string name;
    std::string extension;
    unsigned size;
```

```
};

MultimediaFile::MultimediaFile(const std::string& init_name, const std::string&
init_extension, const unsigned& init_size): name(init_name),
extension(init_extension), size(init_size) {}

void MultimediaFile::set_name(const std::string& new_name){
    this->name = new_name;
}

void MultimediaFile::set_extension(const std::string& new_extension){
    this->extension = new_extension;
}
void MultimediaFile::set_size(const unsigned& new_size){
    this->size = new_size;
}

void MultimediaFile::play(){
    std::cout << this->name << " " << this->extension << " " << this->size;
}

std::string MultimediaFile::get_name() const{
    return this->name;
}
std::string MultimediaFile::get_extension() const{
    return this->extension;
}
unsigned MultimediaFile::get_size() const{
    return this->size;
}

class AudioFile: public MultimediaFile{
public:
    AudioFile() = default;
    AudioFile(const std::string& init_name, const std::string& init_extension,
const unsigned& init_size, const std::string& init_artist, const unsigned&
init_length);

    void set_length(const unsigned& new_length);
    void set_artist(const std::string& new_artist);

    void play() override;

    unsigned get_length() const;
    std::string get_artist() const;
private:
    unsigned length;
    std::string artist;
};

AudioFile::AudioFile(const std::string& init_name, const std::string&
init_extension, const unsigned& init_size, const std::string& init_artist, const
unsigned& init_length)
                    : MultimediaFile(init_name, init_extension, init_size),
length(init_length), artist(init_artist) {}

void AudioFile::set_length(const unsigned& new_length){
    this->length = new_length;
```

```cpp
}
void AudioFile::set_artist(const std::string& new_artist){
    this->artist = new_artist;
}

void AudioFile::play(){
    MultimediaFile::play();
    std::cout << " " << this->length << " " << this->artist;
}

unsigned AudioFile::get_length() const{
    return this->length;
}
std::string AudioFile::get_artist() const{
    return this->artist;
}

class VideoFile: public MultimediaFile{
public:
    VideoFile() = default;
    VideoFile(const std::string& init_name, const std::string& init_extension,
const unsigned& init_size, const std::string& init_subs, const
std::vector<AudioFile>& init_tracks);

    void set_subtitles(const std::string& new_subtitles);
    void add_track(const AudioFile& new_track);

    void play() override;

    std::string get_subtitles() const;
private:
    std::string subtitles;
    std::vector<AudioFile> tracks;
};

VideoFile::VideoFile(const std::string& init_name, const std::string&
init_extension, const unsigned& init_size, const std::string& init_subs, const
std::vector<AudioFile>& init_tracks)
    : MultimediaFile(init_name, init_extension, init_size),
subtitles(init_subs), tracks(init_tracks) {}

void VideoFile::set_subtitles(const std::string& new_subtitles){
    this->subtitles = new_subtitles;
}
void VideoFile::add_track(const AudioFile& new_track){
    this->tracks.push_back(new_track);
}

void VideoFile::play(){
    MultimediaFile::play();
    std::cout << this->subtitles << " ";

    for(auto track: this->tracks){
        track.play();
        std::cout << " ";
    }
}
```

```cpp
std::string VideoFile::get_subtitles() const{
    return this->subtitles;
}
std::vector<AudioFile> VideoFile::get_tracks() const{
    return this->tracks;
}

class Stream: public MultimediaFile{
public:
    Stream() = default;
    Stream(const std::string& init_source);

    void set_source(const std::string& new_source);
    void play() override;
    std::string get_source() const;
private:
    std::string source;
};

Stream::Stream(const std::string& init_source): MultimediaFile("", "", 0),
source(init_source) {}

void Stream::set_source(const std::string& new_source){
    this->source = new_source;
}
void Stream::play(){
    std::cout << this->source;
}
std::string Stream::get_source() const{
    return this->source;
}

class MultimediaPlayer{
public:
    void add_audio_file(const AudioFile& new_audio);
    void add_video_file(const VideoFile& new_video);
    void add_stream(const Stream& new_stream);

    void remove_at(const int& index);

    void play(const int& index);

    ~MultimediaPlayer();
private:
    MultimediaPlayer() = default;
    std::vector<MultimediaFile*> files;
};

void MultimediaPlayer::add_audio_file(const AudioFile& new_audio){
    this->files.push_back(new AudioFile(new_audio));
}
void MultimediaPlayer::add_video_file(const VideoFile& new_video){
    this->files.push_back(new VideoFile(new_video));
}
void MultimediaPlayer::add_stream(const Stream& new_stream){
    this->files.push_back(new Stream(new_stream));
}
```

```
void MultimediaPlayer::remove_at(const int& index){
    this->files.erase(this->files.begin() + index);
}

void MultimediaPlayer::play(const int& index){
    if(dynamic_cast<AudioFile*>(this->files[index]) != nullptr){
        int temp;
        std::cin >> temp;

        if(temp > 0){
            std::cout << "Foobarbaz";
        }
    }
    this->files[index]->play();
}

MultimediaPlayer::~MultimediaPlayer(){
    for(auto* item: this->files){
        delete item;
    }
}
```

Клас `MutimediaFile` (20 точки):

- Виртуален деструктор - 5т
- Виртуален метод `play` - 5т
- get/set (всяко по 1т) - 6т
- Стил - const, const&, unsigned, конструктор с параметри (всяко по 1т) - 4т

Клас `AudioFile` (10 точки):

- Наследява `MultimediaFile` - 5т
- Преизползване на `play` - 2,5т
- get/set (всяко по 0,25) - 1т
- Стил - const, const&, unsigned (всяко по 0,5) - 1,5т

Клас `VideoFile` (15 точки):

- Наследява `MultimediaFile` - 5т
- Колекция от `AudioFile` - 5т
- Преизползване на `play` - 3т
- get/set (всяко по 0,25 + 0,25 за липса на get_tracks) - 1т
- Стил - const, const& (всяко по 0,5) - 1т

Клас `Stream` (10 точки):

- Наследява `MultimediaFile` - 5т
- Член-данните от `MultimediaFile` са празни и не се показват при `play` (всяко по 1,5т) - 3т
- get/set (всяко по 0,5) - 1т
- Стил - const, const& (всяко по 0,5) - 1т

Клас `MultimediaPlayer` (25 точки):

- Хетерогенен контейнер (вектор от указатели - 5т, копиране в динамичната памет - 5т, правилен деструктор - 5т) - 15т
- Правилен метод `play` - 5т
- Методи за добавяне и премахване (всяко по 1т) - 2т
- Стил - const, const&, използване на итератор при `remove` (всяко по 1т) - 3т