```cpp
#include <iostream>
#include <fstream>
#include <Windows.h>
#include <cstring>
#include <string>

using MARKS = short[5];

const int m = 35;

struct STUDENT
{
private:
	char FIO[m];
	short course, group;
	MARKS marks;
public:
	STUDENT() {}
	STUDENT(std::ifstream &file);
	void print();
	int compare(const STUDENT &student);
	int kind();
	void set_FIO(char FIO[])
	{
		strcpy_s(this->FIO, m, FIO);
	}
	short get_course()
	{
		return course;
	}
};

using ptrSTUDENT = STUDENT*;

struct ARRAY
{
	ptrSTUDENT* arr;
	size_t size;
	ARRAY() {};
	ARRAY(std::ifstream &file);
	void print();
	void sorting();
	bool to_binary(const char* file_name);
	~ARRAY()
	{
		for (size_t i = 0; i < size; ++i)
			delete arr[i];
		delete[] arr;
	}
};
```

```
bool print_binary_file(const char* file_name)
{
	bool result = true;
	std::ifstream file(file_name, std::ios::binary);
	if (!file)
		result = false;
	else
	{
		STUDENT student;
		//while (!file.eof())
		//{
		//	file.read((char*)(&student), sizeof(student));
		//	if (!file.fail())
		//		student.print();
		//}
		while (file.read((char*)(&student), sizeof(student)))
		{
			student.print();
		}
		file.close();
	}
	return result;
}

std::string course_excellent(std::ifstream& file_bin)
{
	std::string result = "";
	STUDENT student;
	short cur_course = 0;
	int count = 0, max = 0;
	auto lambda_max = [&]()
	{
		if (count > max)
		{
			max = count;
			result = std::to_string(cur_course);
		}
		else
			if (count == max)
				result += ", " + std::to_string(cur_course);
	};
	while (file_bin.read((char*)(&student), sizeof(student)))
	{
		if (student.get_course() == cur_course)
		{
			//if (student.kind() == 5)
			//	count++;
			count += student.kind() == 5 ? 1 : 0;
		}
		else
		{
			lambda_max();

			cur_course = student.get_course();
			count = student.kind() == 5 ? 1 : 0;
		}
	}
```

```
	file_bin.close();

	lambda_max();

	return result;
}

int check_file(std::ifstream &file)
{
	int result = 1;
	if (!file)
		result = -1;
	else
		if (file.peek() == EOF)
			result = 0;
	return result;
}

int main()
{
	SetConsoleCP(1251);
	SetConsoleOutputCP(1251);
	std::ifstream file("students.txt");
	switch (check_file(file))
	{
	case -1:
		std::cout << "Файл не найден!\n";
		break;
	case 0:
		std::cout << "Файл пуст!\n";
		break;
	default:
	{
		ARRAY array(file);
		array.sorting();
		array.print();
		array.to_binary("data_bin.txt");
		std::cout << "<><><><><> binary file <><><><><>\n";
		print_binary_file("data_bin.txt");

		std::ifstream file_bin("data_bin.txt", std::ios::binary);
		std::string courses = course_excellent(file_bin);
		if (courses == "")
			std::cout << "Отличников нет\n";
		else
			std::cout << "На " << courses << " курсе(ах) больше всего отличников\n";
		break;
	}
	}

	std::cin.get();
	return 0;
}
```

```
STUDENT::STUDENT(std::ifstream & file)
{
	file.getline(FIO, m);
	file >> this->course >> group;
	for (int i = 0; i < 5; i++)
		file >> marks[i];
	file.ignore();
	if (!file.eof())
	{
		char delim_line[10];
		file.getline(delim_line, 10);
	}
}

void STUDENT::print()
{
	std::cout << FIO << '\n';
	std::cout << course << ' ' << group << std::endl;
	for (int i = 0; i < 5; i++)
		std::cout << marks[i] << ' ';
	std::cout << "\n--------------------\n";
}

int STUDENT::compare(const STUDENT & student)
{
	int result = -1;
	if (course > student.course ||
		course == student.course && group>student.group ||
		course == student.course && group == student.group &&
		strcmp(FIO, student.FIO) > 0)
		result = 1;
	else
		if (course == student.course && group == student.group &&
		   strcmp(FIO, student.FIO) == 0)
			result = 0;
	return result;
}

int STUDENT::kind()
{
	int min = marks[0];
	for (int i = 1; i < 5; i++)
		if (marks[i] < min)
			min = marks[i];
	return min;
}

ARRAY::ARRAY(std::ifstream & file)
{
	file >> size;
	file.ignore();
	arr = new ptrSTUDENT[size];
	for (size_t i = 0; i < size; i++)
		arr[i] = new STUDENT(file);
	file.close();
}
```

```
void ARRAY::print()
{
	for (size_t i = 0; i < size; i++)
		arr[i]->print();
}

void ARRAY::sorting()
{
	for (size_t right = size; right >= 2; right--)
		for (size_t i = 0; i < right - 1; i++)
			if (arr[i]->compare(*arr[i + 1])>0)
				std::swap(arr[i], arr[i + 1]);
}

bool ARRAY::to_binary(const char * file_name)
{
	bool result = true;
	std::ofstream file(file_name, std::ios::binary);
	if (!file.is_open())
		result = false;
	else
	{
		for (size_t i = 0; i < size; i++)
			//file.write((char*)&(*arr[i]), sizeof(STUDENT));
			file.write((char*)arr[i], sizeof(STUDENT));

		file.close();
	}
	return result;
}
```