2) Develop a Student Information System in C++ using OOP concepts, including constructor, destructor, static member functions, friend class, dynamic memory allocation, and exception handling

***CODE :-***

#include <iostream>

#include <vector>

#include <cstring>

using namespace std;

class Student

{

static int count;

int roll\_no;

string name, class\_name, division, dob, blood\_group, contact\_address, phone\_no, dl\_no;

public:

Student()

{

roll\_no = 0;

name = class\_name = division = dob = blood\_group = contact\_address = phone\_no = dl\_no = "";

}

Student(int roll, string nm, string cls, string div, string date, string blood, string address, string phone, string dl)

: roll\_no(roll), name(nm), class\_name(cls), division(div), dob(date), blood\_group(blood), contact\_address(address), phone\_no(phone), dl\_no(dl) {

count++;

}

Student(const Student &s)

{

roll\_no = s.roll\_no;

name = s.name;

class\_name = s.class\_name;

division = s.division;

dob = s.dob;

blood\_group = s.blood\_group;

contact\_address = s.contact\_address;

phone\_no = s.phone\_no;

dl\_no = s.dl\_no;

count++;

}

~Student()

{

count--;

}

inline void display() const

{

cout << "Roll No: " << roll\_no << "\nName: " << name << "\nClass: " << class\_name

<< "\nDivision: " << division << "\nDate of Birth: " << dob << "\nBlood Group: " << blood\_group

<< "\nContact Address: " << contact\_address << "\nPhone No: " << phone\_no << "\nDriving License No: " << dl\_no << endl;

}

static int getCount()

{

return count;

}

static Student\* createStudent(int roll, string nm, string cls, string div, string date, string blood, string address, string phone, string dl) {

try

{

Student\* student = new Student(roll, nm, cls, div, date, blood, address, phone, dl);

return student;

}

catch (bad\_alloc &e)

{

cerr << "Memory allocation failed: " << e.what() << endl;

return nullptr;

}

}

static void deleteStudent(Student\* student)

{

delete student;

}

};

int Student::count = 0;

int main()

{

vector<Student\*> students;

try

{

students.push\_back(Student::createStudent(1, "John Doe", "12th", "A", "2005-03-15", "O+", "123 St, City", "1234567890", "DL123"));

students.push\_back(Student::createStudent(2, "Jane Smith", "11th", "B", "2006-07-22", "A-", "456 Ave, City", "9876543210", "DL456"));

for (auto& student : students)

{

student->display();

cout << "--------------------------" << endl;

}

cout << "Total number of students: " << Student::getCount() << endl;

}

catch (exception &e)

{

cerr << "Exception: " << e.what() << endl;

}

for (auto& student : students)

{

Student::deleteStudent(student);

}

return 0;

}

***OUTPUT :-***

