Part (B): A piece of code and its output is given below where a Person has a Date of Birth (DOB). Recall the properties of Composition that we studied in class. **Your task is to modify the given code to implement properties of Composition between Person and DOB using Date* DOB.** *You are **NOT ALLOWED** to have Date object (Date DOB) in Person class.* If required, you may add as many functions or code as you want to fulfil this task. *In your answer, you are required to write only modified/newly added functions/code (if any). You are not required to re-write the function(s) which will not change.* Properly write the functions/code (if any) to get credit. (Note: There is no syntax error in the given code.) Output should remain the same after modification(s).

```
#include<string>
#include<iostream>
using namespace std;

class Date{
    int Day, Month, Year;
public:
    Date(int d=1, int m=1, int y=1960): Day(d), Month(m), Year(y)
    {
    }
    void Print(){cout<<Day<<"-"<<Month<<"-"<<Year;}
};
class Person{
    string Name;
    Date* DOB; //Date of Birth
public:
    Person(string name, Date dob): DOB(new Date(dob))
    {
        Name = name;
        DOB = 0;
    }
    void Print(){
        cout<<"Name: "<<Name<<endl;
        if (DOB)
            cout<<"DOB: "; DOB->Print(); cout<<endl;
    }
    void SetDOB(Date* dptr){DOB = dptr;}
};
int main() {
    Date dob(18,5,2005);
    Person p1("Dummy Employee");
    p1.SetDOB(&dob);

    p1.Print();
    return 0;
}
```

```
Output:
Name: Dummy Employee
DOB: 18-5-2005
Press any key to continue . . .
```

~Person() { } (2)
delete DOB;

CLO # 1: Demonstrate the basic concepts of OOP

Q3: [8+7 = 15 marks]

Part (A): You are given a driver program (main) along with partial implementation of required classes. Expected output of the driver program is also given. **Your task is to write additional function(s) required (if any) such that the driver successfully gives expected output.** Do not re-write already given code. Clearly mention all the changes required. You are also not allowed to change main program. (Note: There is no syntax error in the given code.)

Partial Code:

```
#include<string>
#include<iostream>
using namespace std;

class Date{
      int Day, Month, Year;
public:
      Date(int d=1, int m=1, int y=1960): Day(d), Month(m), Year(y)
      {
      }
      void Print(){cout<<Day<<"-"<<Month<<"-"<<Year;}
};
class Employee{
      string Name;
      Date DOB;
      Date JoiningDate;
public:
      Employee(string name="", int d1=1, int m1=1, int y1=1960, int d2=1, int m2=1, int
y2=1960): Name(name), DOB(d1,m1,y1), JoiningDate(d2, m2, y2)
      {
      }
      void Print(){
            cout<<"Name: "<<Name<<endl;
            cout<<"DOB: "; DOB.Print(); cout<<endl;
            cout<<"Joining Date: "; JoiningDate.Print(); cout<<endl;
      }
};
void main()
{
      Employee emp("Abc Xyz", 29, 12, 2000, 10, 6, 2020);
      emp.Print();

      emp["DOB"][0] = 15;
      emp["JoiningDate"][2] = 2022;

      cout<<"--------------------"<<endl;
      emp.Print();
}
```

int & operator[](int i)
if(i==0) return Day
else if(i==1) return Month
else year;

Date & operator[](string str)
if(str=="DOB")
return DoB;
else
joinDate;

Expected Output:

```
Name: Abc Xyz
DOB: 29-12-2000
Joining Date: 10-6-2020
--------------------
Name: Abc Xyz
DOB: 15-12-2000
Joining Date: 10-6-2022
Press any key to continue . . .
```

You are given a main function and its required output. **Best using the practices studied in the class, write missing functionality such that this program runs successfully and gives required output.** You are NOT ALLOWED to change main program and there is no syntax error in this code. *No credit will be given for bad solution.*

Main Program

```
#include<iostream>
using namespace std;
int main() {
    int intArr[] = { 5, 2, 9, 1, 6 };
    int intSize = sizeof(intArr) / sizeof(int);
    MySort(intArr, intSize);
    cout << "Sorted integer array:\n";
    for (int i = 0; i < intSize; ++i) {
        cout << intArr[i] <<endl;
    }

    float floatArr[] = { 5.2, 9.7, 2.6, 6.5, 1.9 };
    int floatSize = sizeof(floatArr) / sizeof(float);
    MySort(floatArr, floatSize);
    cout << "Sorted floats array:\n";
    for (int i = 0; i < floatSize; ++i) {
        cout << floatArr[i] <<endl;
    }

    const char* strArr[] = { "banana", "apple", "orange",
"grape", "kiwi" };
    int strSize = sizeof(strArr) / sizeof(char*);
    MySort(strArr, strSize);
    cout << "Sorted string array:\n";
    for (int i = 0; i < strSize; ++i) {
        cout << strArr[i] <<endl;
    }
    return 0;
}
```

Required Output

```
Sorted integer array:
1
2
5
6
9
Sorted floats array:
1.9
2.6
5.2
6.5
9.7
Sorted string array:
apple
banana
grape
kiwi
orange
Press any key to continue . . .
```

428

There is no syntax error in this code.)

```
id SomeFunction(A& obj)

    obj.func2();

id AnotherFunction(A& obj)

    obj.func();

 main()

    B b;
    b.func();
    SomeFunction(b);
    AnotherFunction(b);
    A a_obj2 = b;
    a_obj2.func();
    return 0;
```

A B C
4 5 6
A B C
1 2 3

1.5 each

agar last wali lay swapped hon to -3.5

g the practices studied in the class, write missing required output. You are NOT ALLOWED to change be given for bad solution.

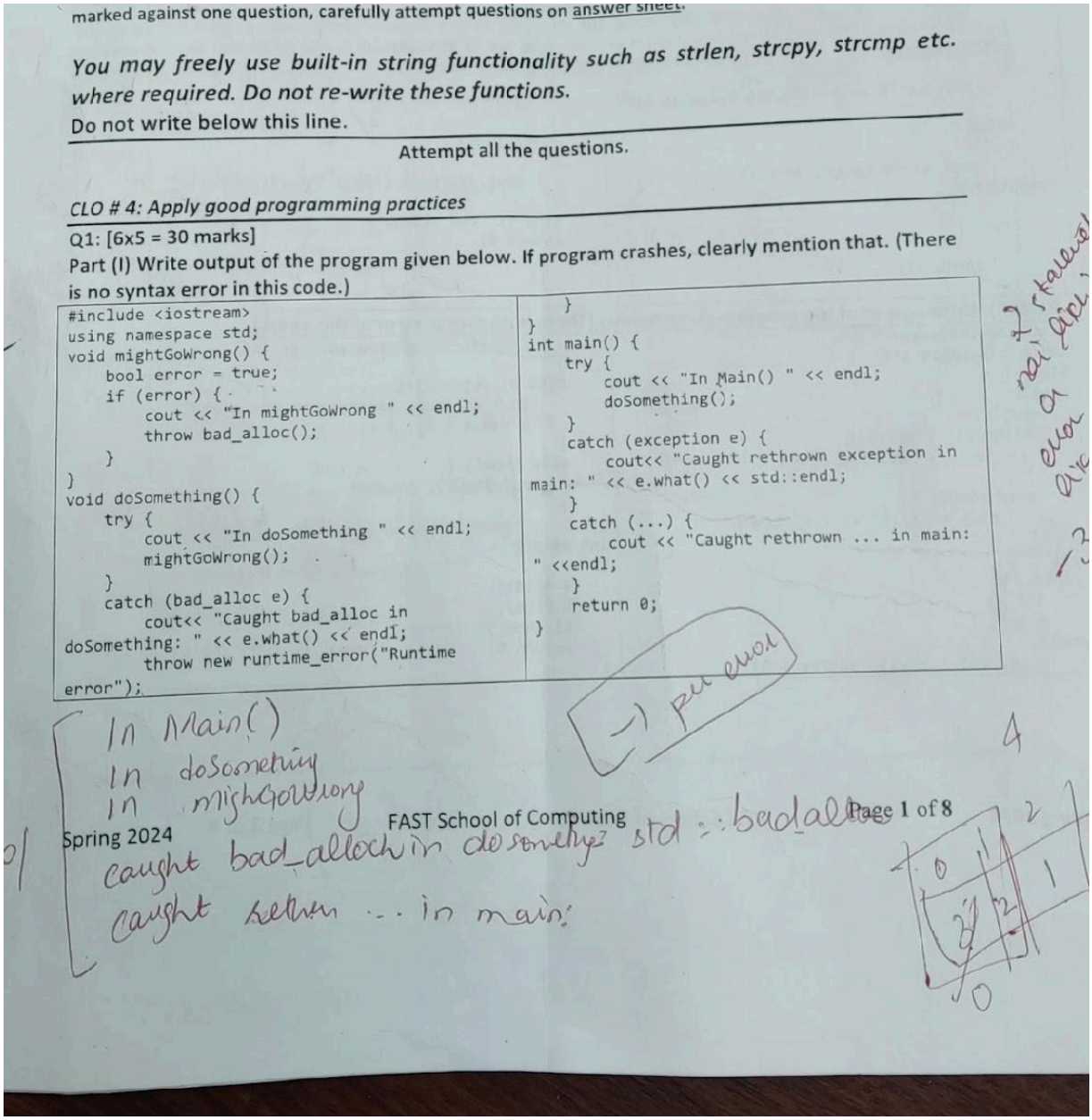
	Required Output
	Sorted integer array: 1 2 5 6 9 Sorted floats array: 1.9 2.6 5.2 6.5 9.7 Sorted string array: apple banana grape kiwi orange Press any key to continue . . .

Part (II) Write output of the program given below. If program crashes, clearly mention that. (There is no syntax error in this code.)

```
#include<iostream>
using namespace std;
void D()
{
    cout << "Start D\n";
    cout << "D throwing int exception\n";
    throw - 1;
    cout << "End D\n";
}
void C()
{
    cout << "Start C\n";
    D();
    cout << "End C\n";
}
void B()
{
    cout << "Start B\n";
    try
    {
        C();
    }
    catch (double)
    {
        cout << "B caught double exception\n";
    }
    catch (...)
    {
        cout << "B caught default
exception\n";
    }
    try
    {
        throw -1;
    }
    catch (int)
    {
        cout << "B caught int exception\n";
    }
    cout << "End B\n";
}
void A()
{
    cout << "Start A\n";
    try
    {
        B();
    }
    catch (int)
    {
        cout << "A caught int exception\n";
    }
    cout << "End A\n";
}
int main()
{
    cout << "Start main\n";
    try
    {
        A();
    }
    catch (int)
    {
        cout << "main caught int exception\n";
    }
    cout << "End main\n";
    return 0;
}
```

Part(III) Write output of the program given below. (There is no syntax error in this code.)

```
#include<iostream>
using namespace std;
class A {
public:
    int i;
    A(int ii) : i(ii) {
        cout << "Calling A(int ii)" << endl;
    }
    void show() {
        cout <<"A i=" << i << endl;
    }
};
class B {
    int x;
    A obj;
public:
    B(int xx) : x(xx), obj(xx + 5) {
        cout << "B constructor" << endl;
    }
    B(B& o):obj(o.x) {
        cout << "Copy constructor B" << endl;
        x = o.x + 5;
    }
    void show() {
        obj.show();
    }
};
int main()
{
    B b(10);
    B b1(b);
    b1.show();
    return 0;
}
```



marked against one question, carefully attempt questions on answer sheet.

You may freely use built-in string functionality such as strlen, strcpy, strcmp etc. where required. Do not re-write these functions.

Do not write below this line.

Attempt all the questions.

CLO # 4: Apply good programming practices

Q1: [6x5 = 30 marks]

Part (I) Write output of the program given below. If program crashes, clearly mention that. (There is no syntax error in this code.)

```
#include <iostream>
using namespace std;
void mightGoWrong() {
    bool error = true;
    if (error) {
        cout << "In mightGoWrong " << endl;
        throw bad_alloc();
    }
}
void doSomething() {
    try {
        cout << "In doSomething " << endl;
        mightGoWrong();
    }
    catch (bad_alloc e) {
        cout<< "Caught bad_alloc in
doSomething: " << e.what() << endl;
        throw new runtime_error("Runtime
error");
    }
}
int main() {
    try {
        cout << "In Main() " << endl;
        doSomething();
    }
    catch (exception e) {
        cout<< "Caught rethrown exception in
main: " << e.what() << std::endl;
    }
    catch (...) {
        cout << "Caught rethrown ... in main:
" <<endl;
    }
    return 0;
}
```


Last Question 4 Solution:

```
#include <iostream>

#include <string>
```

```
using namespace std;

// Utility functions (provided as given)
char* GetAddressType(int type) {
   if (type == 1) return "Home";
   else if (type == 2) return "Work";
   else return "Other";
}

char* GetPhoneType(int type) {
   if (type == 1) return "Mobile";
   else if (type == 2) return "Home";
   else if (type == 3) return "Work fax";
   else return "Other";
}

char* StringDeepCopy(const char* src) {
   char* dest = new char[strlen(src) + 1];
   strcpy(dest, src);
   return dest;
}

// Abstract base class for contact details
class ContactDetail {
protected:
   int type;
public:
   ContactDetail(int t) : type(t) {}
   virtual void Print() = 0;
```

```
    virtual ~ContactDetail() {}
};

// Derived class for phone numbers
class PhoneNo : public ContactDetail {
    char* number;
public:
    PhoneNo(int t, const char* num) : ContactDetail(t) {
        number = StringDeepCopy(num);
    }
    void Print() override {
        cout << "[" << GetPhoneType(type) << "] " << number << endl;
    }
    ~PhoneNo() {
        delete[] number;
    }
};

// Derived class for addresses
class Address : public ContactDetail {
    char* street;
    char* city;
    char* state;
    char* postcode;
    char* country;
public:
    Address(int t, const char* str, const char* cty, const char* st, const char* pc, const char* cntry) :
ContactDetail(t) {
        street = StringDeepCopy(str);
```

```
        city = StringDeepCopy(cty);
        state = StringDeepCopy(st);
        postcode = StringDeepCopy(pc);
        country = StringDeepCopy(cntry);
    }
    void Print() override {
        cout << "[" << GetAddressType(type) << "] " << street << endl;
        cout << city << "," << state << " " << postcode << endl;
        cout << country << endl;
    }
    ~Address() {
        delete[] street;
        delete[] city;
        delete[] state;
        delete[] postcode;
        delete[] country;
    }
};

// Class for managing contacts
class Contact {
    string name;
    ContactDetail** details;
    int detailCount;
public:
    Contact(const string& n) : name(n), details(nullptr), detailCount(0) {}

    void AddContactDetail(ContactDetail* detail) {
        ContactDetail** newDetails = new ContactDetail*[detailCount + 1];
```

```
        for (int i = 0; i < detailCount; i++) {
            newDetails[i] = details[i];
        }
        newDetails[detailCount] = detail;
        delete[] details;
        details = newDetails;
        detailCount++;
    }

    void Print() {
        cout << "[Name] " << name << endl;
        for (int i = 0; i < detailCount; i++) {
            details[i]->Print();
        }
    }

    ~Contact() {
        for (int i = 0; i < detailCount; i++) {
            delete details[i];
        }
        delete[] details;
    }
};

// Main function demonstrating usage
int main() {
    Contact contact1("Ali Hamza");
    contact1.Print();
    cout << "--------------------------------" << endl;
```

```
    Contact contact2("Usman Khalid");
    contact2.Print();
    cout << "--------------------------------" << endl;

    contact1.AddContactDetail(new PhoneNo(1, "0300-1234567"));
    contact1.Print();
    cout << "--------------------------------" << endl;

    contact1.AddContactDetail(new PhoneNo(3, "042-111-128-128"));
    contact1.AddContactDetail(new Address(2, "852-B Milaad St, Block B Faisal Town", "Lahore", "Punjab", "54770", "Pakistan"));
    contact1.AddContactDetail(new Address(1, "853-B Faisal Town", "Lahore", "Punjab", "54770", "Pakistan"));
    contact2.AddContactDetail(new Address(3, "123-B Xyz Town", "Gujranwala", "Punjab", "12345", "Pakistan"));

    contact1.Print();
    cout << "--------------------------------" << endl;
    contact2.Print();
    cout << "--------------------------------" << endl;

    return 0;
}
```

Q2:

```
#include<iostream>
```

```
#include<fstream>
using namespace std;

//Partial Definition of class Video
class Video{
	int* TagsList;
	int ID;
public:
	int* GetTagsList() { return TagsList; }
	int GetID() { return ID; }
	//Other functions...
	Video(){TagsList = 0; ID = 0;}
	void SetValues(ifstream& fin){
		int tags = 0;
		fin>>ID;
		fin>>tags;
		TagsList = new int[tags];
		char ch;
		int i=0;
		for(; i<tags-1; i++)
		{
			fin>>TagsList[i];
			fin>>ch;
		}
		fin>>TagsList[i];
	}
	~Video()
	{
		if(TagsList) delete[] TagsList;
```

```
	}
	void Print(){
		for(int i=0; TagsList[i] != -1 ; i++)
			cout<<TagsList[i]<<"\t";
		cout<<endl;
	}
};
//Partial Definition of class VideoSystem
class VideoSystem{
	Video* allVideos;
	int totalVideos;
public:
	//Other Functions...
	VideoSystem(){
		ifstream fin("Data2.txt");
		if(fin.is_open())
		{
			fin>>totalVideos;
			allVideos = new Video[totalVideos];
			for(int i=0; i<totalVideos; i++)
			{
				allVideos[i].SetValues(fin);
			}
			for(int i=0; i<totalVideos; i++)
			{
				cout<<"Video "<<i<<":\t";
				allVideos[i].Print();
			}
		fin.close();
```

```
        }
}
~VideoSystem()
{
        if(allVideos)
                delete[] allVideos;
}
bool PrefFound(int* videoTags, int userPref)
{
        for(int i=0; videoTags[i] != -1 ; i++)
        {
                if(videoTags[i] == userPref)
                        return true;
        }
        return false;

}
bool VideoMatchesUserPref(int* videoTags, int*& userPref)
{
        int count = 0;
        for(int i=0; userPref[i]!= -1; i++)
        {
                if(PrefFound(videoTags, userPref[i]))
                {
                        count++;
                        if(count == 2)
                                return true;
                }
        }
```

```
            return false;
        }
        int* GetRecommendedVideos(int* userPref)
        {
            int* recommendedVideos = new int[totalVideos];
            int j=0;
            for(int i=0; i<totalVideos; i++)
            {
                if(VideoMatchesUserPref( allVideos[i].GetTagsList(), userPref))
                {
                    recommendedVideos[j++] = allVideos[i].GetID();
                }
            }
            recommendedVideos[j] = -1;
            return recommendedVideos;
        }
};

void main()
{
        VideoSystem videoSystem;
        int userPref[] = {1,5,9,-1};

        int* recommendations = videoSystem.GetRecommendedVideos(userPref);
        cout<<"Suggested Videos:\t";
        for(int i=0; recommendations[i] != -1 ; i++)
        {
            cout<<recommendations[i]<<"\t";
        }
```

```
        cout<<endl;
}
/*
Data File:
5
49516
8
1,2,3,4,6,7,8,-1
241793
4
1,3,5,-1
284957
3
1,6,-1
123456
4
2,5,9,-1
654321
6
3,4,5,6,9,-1
*/
```