

מבחן מועד Y - שאלון

במבחן 10 שאלות. משקל כל שאלה הוא 10 נקודות.

עבור כל שאלה יש להתייחס לדברים הבאים:

- ✓ במידה והיא מתקמפלת יש לכתוב מהו הפלט, כולל מה שיודפס לאחר הסוגריים המסולסלים שסוגרים את התוכנית
- ✓ במידה והשאלה אינה מתקמפלת, יש לכתוב מדוע
- ✓ במידה וישנה תעופה מהקוד, יש לכתוב מהו הפלט עד התעופה ולהסביר מדוע הקוד עף
- ✓ במידה ויש קוד שלא כל הזיכרון שוחרר, יש לציין זאת

שאלות 1 עד 5 יתייחסו לקטע הקוד הבא:

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

template<class T>
class A
{
    class A_Impl
    {
        T x;
        int version;
    public:
        A_Impl(T x) : x(x), version(1)
        {
            cout << "Creating A_Impl x=" << x << endl;
        }

        A_Impl(const A_Impl& a) : x(a.x), version(1)
        {
            cout << "Copying A_Impl x=" << x << endl;
        }

        A_Impl(A_Impl&& a) : x(a.x), version(1)
        {
            cout << "Moving A_Impl x=" << x << endl;
        }

        const A_Impl& operator=(const A_Impl& a)
        {
            x = a.x;
            ++version;
            return *this;
        }

        operator T()const { return x; }
        friend class A<T>;
    };

    A_Impl a;

public:
    A(T x) : a(x) {}
    A_Impl& get() { cout << "get ver1\n"; return a; }
    const A_Impl& get() const { cout << "get ver2\n"; return a; }
    int version() { return a.version; }
};

int getImplValue(const A<int>& a)
{
    return a.get();
}
```

```

void main1()
{
    A<int> a1(5);
    cout << "a1 = " << a1.get() << ", version = " << a1.version() << endl;
    cout << "1 -----\\n";

    a1.get() = 7;
    cout << "a1 = " << a1.get() << ", version = " << a1.version() << endl;
    cout << "2 -----\\n";

    A<int> a2 = a1;
    a1 = 13;
    cout << "a1 = " << a1.get() << ", version = " << a1.version() << endl;
    cout << "a2 = " << a2.get() << ", version = " << a2.version() << endl;
    cout << "3 -----\\n";

    cout << getImplValue(A<int>(a1)) << endl;
    cout << "4 -----\\n";
}

void main2()
{
    A<A<int>> aa(A<int>(8));
    cout << "aa = " << aa.get() << ", version = " << aa.version() << endl;
}

```

שאלה 1:

עבור main1 - מה יהיה הפלט עד קו 1?

שאלה 2:

עבור main1 - מה יהיה הפלט בין קו 1 לקו 2?

שאלה 3:

עבור main1 - מה יהיה הפלט בין קו 2 לקו 3?

שאלה 4:

עבור main1 - מה יהיה הפלט בין קו 3 לקו 4?

שאלה 5:

מה יקרה עבור main2?

שאלות 6 עד 10 יתייחסו לקטע הקוד הבא:

```
#pragma warning(disable: 4996)

#include <iostream>
using namespace std;

#include <string>
#include <vector>

class IceCream;

class IceCreamCombination
{
public:
    virtual void toOs(ostream& os) const = 0;

    friend ostream& operator<<(ostream& os, const IceCreamCombination& s)
    {
        s.toOs(os);
        return os;
    }

    virtual IceCreamCombination* clone() const = 0;
};

class IceCream : public IceCreamCombination
{
    char* flavour;
public:
    IceCream(const string& flavour)
    {
        cout << "Creating " << flavour << " ice-cream\n";
        this->flavour = new char[flavour.size() + 1];
        strcpy(this->flavour, flavour.c_str());
    }

    IceCream(const IceCream& other)
    {
        this->flavour = strdup(other.flavour);
        cout << "Copying " << flavour << " ice-cream\n";
    }

    IceCream(IceCream&& other) : flavour(nullptr)
    {
        swap(this->flavour, other.flavour);
        cout << "Moving " << flavour << " ice-cream\n";
    }

    ~IceCream()
    {
        cout << "Eating " << (flavour ? flavour : "") << " ice-cream\n";
        delete[] flavour;
    }

    void changeFlavour(const char* newFlavour)
    {
        delete[] this->flavour;
        this->flavour = strdup(newFlavour);
    }
}
```

```

    virtual void toOs(ostream& os) const override
    {
        os << flavour << " ice cream";
    }

    operator const char* () { return flavour; }

    virtual IceCreamCombination* clone() const
    {
        return new IceCream(*this);
    }
};

class IceCreamTopping : public IceCreamCombination
{
    IceCreamCombination* theIceCream;
public:
    IceCreamTopping(IceCreamCombination* theIceCream)
        : theIceCream(theIceCream) {}

    IceCreamTopping(const IceCreamTopping& other)
        : IceCreamCombination(other)
    {
        theIceCream = other.theIceCream->clone();
    }

    virtual void toOs(ostream& os) const override
    {
        os << *theIceCream;
    }

    IceCreamCombination* getIceCreamCombination() { return theIceCream; }

    virtual IceCreamCombination* clone() const
    {
        return new IceCreamTopping(*this);
    }
};

class CreamTopping : public IceCreamTopping
{
    int grams;
public:
    CreamTopping(IceCreamCombination* theIceCream, int grams)
        : IceCreamTopping(theIceCream), grams(grams)
    {
        cout << "Adding " << grams << "gr. cream!\n";
    }

    virtual void toOs(ostream& os) const override
    {
        IceCreamTopping::toOs(os);
        os << ", grams=" << grams;
    }

    void addGrams(int grams) { this->grams += grams; }
    int getGrams() const { return grams; }

    virtual IceCreamCombination* clone() const
    {
        return new CreamTopping(*this);
    }
};

```

```

class ChocolateTopping : public IceCreamTopping
{
public:
    enum class eType { MILK, WHITE, DARK };
private:
    eType type;

    static const char* typeStr[3];
public:
    ChocolateTopping(IceCreamCombination* theIceCream, eType type)
        : IceCreamTopping(theIceCream), type(type)
    {
        cout << "Adding topping of " << typeStr[(int)type]
              << " chocolate!\n";
    }

    virtual void toOs(ostream& os) const override
    {
        IceCreamTopping::toOs(os);
        os << ", type=" << typeStr[(int)type];
    }

    virtual IceCreamCombination* clone() const
    {
        return new ChocolateTopping(*this);
    }
};

const char* ChocolateTopping::typeStr[3] = { "Milk", "White", "Dark" };

void main1()
{
    IceCreamCombination* s2 = new CreamTopping(
        new ChocolateTopping(
            new IceCream("vanilla"), ChocolateTopping::eType::MILK), 30);
    cout << *s2 << endl;

    cout << endl;
}

void main2()
{
    vector<IceCream> allIceCreams;
    allIceCreams.reserve(3);
    allIceCreams.push_back(IceCream("chocolate"));
    allIceCreams.push_back(IceCream("Stawberry"));
    CreamTopping ct(&allIceCreams[0], 40);
    cout << ct << endl;
}

```

```

void main3()
{
    vector<IceCreamCombination*> allIceCreams;
    allIceCreams.push_back(new IceCream("vanilla"));
    allIceCreams.push_back(new ChocolateTopping(
        new IceCream("chocolate"), ChocolateTopping::eType::WHITE));
    allIceCreams.push_back(new CreamTopping(new IceCream("chocolate"), 45));

    vector<IceCreamCombination*>::iterator itr = allIceCreams.begin();
    vector<IceCreamCombination*>::iterator itrEnd = allIceCreams.end();
    int sum = 0;
    for (; itr != itrEnd; ++itr)
    {
        CreamTopping* cb = dynamic_cast<CreamTopping*>(*itr);
        if (cb)
            sum += cb->getGrams();
    }

    cout << sum << endl;
}

```

```

void main4()
{
    IceCreamCombination* s1 = new CreamTopping(
        new ChocolateTopping(
            new IceCream("vanilla"),
            ChocolateTopping::eType::MILK), 30);

    cout << "1-----\n";

    CreamTopping* sc1 =
        dynamic_cast<CreamTopping*>(dynamic_cast<IceCreamTopping*>
            (s1)->getIceCreamCombination());
    if (sc1 != nullptr)
    {
        sc1->addGrams(10);
        cout << *sc1 << endl;
    }
    cout << "2 ----- \n";

    ChocolateTopping* sc2 =
        dynamic_cast<ChocolateTopping*>(dynamic_cast<IceCreamTopping*>
            (s1)->getIceCreamCombination());
    if (sc2 != nullptr)
    {
        cout << *sc2 << endl;
    }
    cout << "3 ----- \n";

    CreamTopping* s3 = dynamic_cast<CreamTopping*>(s1->clone());
    IceCreamTopping* temp = dynamic_cast<IceCreamTopping*>
        (s3->getIceCreamCombination());
    IceCream* ic = dynamic_cast<IceCream*>(temp->getIceCreamCombination());
    if (ic)
    {
        ic->changeFlavour("banana");
        cout << *s1 << endl;
        cout << *s3 << endl;
        cout << *ic << endl;
    }

    delete s1;
    delete sc1;
    delete sc2;
    delete s3;
    cout << "4 ----- \n";
}

```


שאלה 6:

מה יקרה בעקבות הרצת הפונקציה main1?

שאלה 7:

מה יקרה בעקבות הרצת הפונקציה main2?

שאלה 8:

מה יקרה בעקבות הרצת הפונקציה main3?

שאלה 9:

מה יקרה בעקבות הרצת הפונקציה main4 עד הקו 3?

שאלה 10:

מה יקרה בעקבות הרצת הפונקציה main4 אחרי הקו 3?

מבחן מועד Y - פתרון

במבחן 10 שאלות. משקל כל שאלה הוא 10 נקודות.

עבור כל שאלה יש להתייחס לדברים הבאים:

- ✓ במידה והיא מתקמפלת יש לכתוב מהו הפלט, כולל מה שיודפס לאחר הסוגריים המסולסלים שסוגרים את התוכנית
- ✓ במידה והשאלה אינה מתקמפלת, יש לכתוב מדוע
- ✓ במידה וישנה תעופה מהקוד, יש לכתוב מהו הפלט עד התעופה ולהסביר מדוע הקוד עף
- ✓ במידה ויש קוד שלא כל הזיכרון שוחרר, יש לציין זאת

שאלות 1 עד 5 יתייחסו לקטע הקוד הבא:

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

template<class T>
class A
{
    class A_Impl
    {
        T x;
        int version;
    public:
        A_Impl(T x) : x(x), version(1)
        {
            cout << "Creating A_Impl x=" << x << endl;
        }

        A_Impl(const A_Impl& a) : x(a.x), version(1)
        {
            cout << "Copying A_Impl x=" << x << endl;
        }

        A_Impl(A_Impl&& a) : x(a.x), version(1)
        {
            cout << "Moving A_Impl x=" << x << endl;
        }

        const A_Impl& operator=(const A_Impl& a)
        {
            x = a.x;
            ++version;
            return *this;
        }

        operator T()const { return x; }
        friend class A<T>;
    };

    A_Impl a;

public:
    A(T x) : a(x) {}
    A_Impl& get() { cout << "get ver1\n"; return a; }
    const A_Impl& get() const { cout << "get ver2\n"; return a; }
    int version() { return a.version; }
};

int getImplValue(const A<int>& a)
{
    return a.get();
}
```

```

void main1()
{
    A<int> a1(5);
    cout << "a1 = " << a1.get() << ", version = " << a1.version() << endl;
    cout << "1 -----\\n";

    a1.get() = 7;
    cout << "a1 = " << a1.get() << ", version = " << a1.version() << endl;
    cout << "2 -----\\n";

    A<int> a2 = a1;
    a1 = 13;
    cout << "a1 = " << a1.get() << ", version = " << a1.version() << endl;
    cout << "a2 = " << a2.get() << ", version = " << a2.version() << endl;
    cout << "3 -----\\n";

    cout << getImplValue(A<int>(a1)) << endl;
    cout << "4 -----\\n";
}

void main2()
{
    A<A<int>> aa(A<int>(8));
    cout << "aa = " << aa.get() << ", version = " << aa.version() << endl;
}

```

שאלה 1:

עבור main1 - מה יהיה הפלט עד קו 1?

פתרון:

```

Creating A_Impl x=5
get ver1
a1 = 5, version = 1

```

שאלה 2:

עבור main1 - מה יהיה הפלט בין קו 1 לקו 2?

פתרון:

```

Creating A_Impl x=7
get ver1
get ver1
a1 = 7, version = 2

```

שאלה 3:

עבור main1 - מה יהיה הפלט בין קו 2 לקו 3?

פתרון:

```

Copying A_Impl x=7
Creating A_Impl x=13
get ver1
a1 = 13, version = 3
get ver1
a2 = 7, version = 1

```

שאלה 4:

עבור main1 - מה יהיה הפלט בין קו 3 לקו 4?

פתרון:

```
Copying A_Impl x=13  
get ver2  
13
```

שאלה 5:

מה יקרה עבור main2?

פתרון:

הקוד לא יתקמפל כי למחלקה Tem אין אופרטור <<

שאלות 6 עד 10 יתייחסו לקטע הקוד הבא:

```
#pragma warning(disable: 4996)

#include <iostream>
using namespace std;

#include <string>
#include <vector>

class IceCream;

class IceCreamCombination
{
public:
    virtual void toOs(ostream& os) const = 0;

    friend ostream& operator<<(ostream& os, const IceCreamCombination& s)
    {
        s.toOs(os);
        return os;
    }

    virtual IceCreamCombination* clone() const = 0;
};

class IceCream : public IceCreamCombination
{
    char* flavour;
public:
    IceCream(const string& flavour)
    {
        cout << "Creating " << flavour << " ice-cream\n";
        this->flavour = new char[flavour.size() + 1];
        strcpy(this->flavour, flavour.c_str());
    }

    IceCream(const IceCream& other)
    {
        this->flavour = strdup(other.flavour);
        cout << "Copying " << flavour << " ice-cream\n";
    }

    IceCream(IceCream&& other) : flavour(nullptr)
    {
        swap(this->flavour, other.flavour);
        cout << "Moving " << flavour << " ice-cream\n";
    }

    ~IceCream()
    {
        cout << "Eating " << (flavour ? flavour : "") << " ice-cream\n";
        delete[] flavour;
    }

    void changeFlavour(const char* newFlavour)
    {
        delete[] this->flavour;
        this->flavour = strdup(newFlavour);
    }
}
```

```

        virtual void toOs(ostream& os) const override
        {
            os << flavour << " ice cream";
        }

        operator const char* () { return flavour; }

        virtual IceCreamCombination* clone() const
        {
            return new IceCream(*this);
        }
    };

    class IceCreamTopping : public IceCreamCombination
    {
    public:
        IceCreamCombination* theIceCream;

        IceCreamTopping(IceCreamCombination* theIceCream)
            : theIceCream(theIceCream) {}

        IceCreamTopping(const IceCreamTopping& other)
            : IceCreamCombination(other)
        {
            theIceCream = other.theIceCream->clone();
        }

        virtual void toOs(ostream& os) const override
        {
            os << *theIceCream;
        }

        IceCreamCombination* getIceCreamCombination() { return theIceCream; }

        virtual IceCreamCombination* clone() const
        {
            return new IceCreamTopping(*this);
        }
    };

    class CreamTopping : public IceCreamTopping
    {
    public:
        int grams;

        CreamTopping(IceCreamCombination* theIceCream, int grams)
            : IceCreamTopping(theIceCream), grams(grams)
        {
            cout << "Adding " << grams << "gr. cream!\n";
        }

        virtual void toOs(ostream& os) const override
        {
            IceCreamTopping::toOs(os);
            os << ", grams=" << grams;
        }

        void addGrams(int grams) { this->grams += grams; }
        int getGrams() const { return grams; }

        virtual IceCreamCombination* clone() const
        {
            return new CreamTopping(*this);
        }
    };

```

```

class ChocolateTopping : public IceCreamTopping
{
public:
    enum class eType { MILK, WHITE, DARK };
private:
    eType type;

    static const char* typeStr[3];
public:
    ChocolateTopping(IceCreamCombination* theIceCream, eType type)
        : IceCreamTopping(theIceCream), type(type)
    {
        cout << "Adding topping of " << typeStr[(int)type]
              << " chocolate!\n";
    }

    virtual void toOs(ostream& os) const override
    {
        IceCreamTopping::toOs(os);
        os << ", type=" << typeStr[(int)type];
    }

    virtual IceCreamCombination* clone() const
    {
        return new ChocolateTopping(*this);
    }
};

const char* ChocolateTopping::typeStr[3] = { "Milk", "White", "Dark" };

void main1()
{
    IceCreamCombination* s2 = new CreamTopping(
        new ChocolateTopping(
            new IceCream("vanilla"), ChocolateTopping::eType::MILK), 30);
    cout << *s2 << endl;

    cout << endl;
}

void main2()
{
    vector<IceCream> allIceCreams;
    allIceCreams.reserve(3);
    allIceCreams.push_back(IceCream("chocolate"));
    allIceCreams.push_back(IceCream("Stawberry"));
    CreamTopping ct(&allIceCreams[0], 40);
    cout << ct << endl;
}

```

```

void main3()
{
    vector<IceCreamCombination*> allIceCreams;
    allIceCreams.push_back(new IceCream("vanilla"));
    allIceCreams.push_back(new ChocolateTopping(
        new IceCream("chocolate"), ChocolateTopping::eType::WHITE));
    allIceCreams.push_back(new CreamTopping(new IceCream("chocolate"), 45));

    vector<IceCreamCombination*>::iterator itr = allIceCreams.begin();
    vector<IceCreamCombination*>::iterator itrEnd = allIceCreams.end();
    int sum = 0;
    for (; itr != itrEnd; ++itr)
    {
        CreamTopping* cb = dynamic_cast<CreamTopping*>(*itr);
        if (cb)
            sum += cb->getGrams();
    }

    cout << sum << endl;
}

```



```

void main4()
{
    IceCreamCombination* s1 = new CreamTopping(
        new ChocolateTopping(
            new IceCream("vanilla"),
            ChocolateTopping::eType::MILK), 30);

    cout << "1-----\n";

    CreamTopping* sc1 =
        dynamic_cast<CreamTopping*>(dynamic_cast<IceCreamTopping*>
            (s1)->getIceCreamCombination());
    if (sc1 != nullptr)
    {
        sc1->addGrams(10);
        cout << *sc1 << endl;
    }
    cout << "2 ----- \n";

    ChocolateTopping* sc2 =
        dynamic_cast<ChocolateTopping*>(dynamic_cast<IceCreamTopping*>
            (s1)->getIceCreamCombination());
    if (sc2 != nullptr)
    {
        cout << *sc2 << endl;
    }
    cout << "3 ----- \n";

    CreamTopping* s3 = dynamic_cast<CreamTopping*>(s1->clone());
    IceCreamTopping* temp = dynamic_cast<IceCreamTopping*>
        (s3->getIceCreamCombination());
    IceCream* ic = dynamic_cast<IceCream*>(temp->getIceCreamCombination());
    if (ic)
    {
        ic->changeFlavour("banana");
        cout << *s1 << endl;
        cout << *s3 << endl;
        cout << *ic << endl;
    }

    delete s1;
    delete sc1;
    delete sc2;
    delete s3;
    cout << "4 ----- \n";
}

```

שאלה 6:

מה יקרה בעקבות הרצת הפונקציה main1?

פתרון:

```
Creating vanilla ice-cream
Adding topping of Milk chocolate!
Adding 30gr. cream!
vanilla ice cream, type=Milk, grams=30
```

שאלה 7:

מה יקרה בעקבות הרצת הפונקציה main2?

פתרון:

```
Creating chocolate ice-cream
Moving chocolate ice-cream
Eating ice-cream
Creating Stawberry ice-cream
Moving Stawberry ice-cream
Eating ice-cream
Adding 40gr. cream!
chocolate ice cream, grams=40
Eating chocolate ice-cream
Eating Stawberry ice-cream
```

שאלה 8:

מה יקרה בעקבות הרצת הפונקציה main3?

פתרון:

```
Creating vanilla ice-cream
Creating chocolate ice-cream
Adding topping of White chocolate!
Creating chocolate ice-cream
Adding 45gr. cream!
45
```

שאלה 9:

מה יקרה בעקבות הרצת הפונקציה main4 עד הקו 3?

פתרון:

```
Creating vanilla ice-cream
Adding topping of Milk chocolate!
Adding 30gr. cream!
-----1
-----2
vanilla ice cream, type=Milk
-----3
```

שאלה 10:

מה יקרה בעקבות הרצת הפונקציה main4 אחרי הקו 3?

פתרון:

```
Copying vanilla ice-cream
vanilla ice cream, type=Milk, grams=30
banana ice cream, type=Milk, grams=30
banana ice cream
```