Programming in Modern C++: Assignment Week 8

Total Marks : 20

Partha Pratim Das
Department of Computer Science and Engineering
Indian Institute of Technology
Kharagpur – 721302
partha.p.das@gmail.com

September 6, 2023

Question 1

Consider the following program. *[MCQ, Marks 2]*

```
#include<iostream>

class AEx{};
class BEx : public AEx{};
class CEx : public BEx{};
void evalEx(int i){
    if(i == 0)
        throw CEx();
    else if(i < 0)
        throw BEx();
    else
        throw AEx();

}
int main(){
    try{
        evalEx(-5);
        evalEx(5);
        evalEx(0);
    }
    catch(int& i){                //LINE-1
        std::cout << "int";
    }
    catch(...){                   //LINE-2
        std::cout << "all";
    }
    catch(BEx& e){                //LINE-3
        std::cout << "BEx" << " ";
    }
    catch(AEx& e){                //LINE-4
        std::cout << "BEx" << " ";
    }
    catch(CEx& e){                  //LINE-5
        std::cout << "CEx" << ", ";
    }
    return 0;
}
```

What will be the output/error?

a) `all`

b) `BEx`

c) Error at `LINE-1`

d) Error at `LINE-2`

Answer: d)
Explanation:
It is an error since '...' handler must be the last handler for its try block.

Question 2

Consider the code segment given below. *[MCQ, Marks 2]*

```
#include<iostream>

namespace Exceptions{
    class AEx{};
    class BEx : public AEx{};
    class CEx : public BEx{};
}
void evalEx(){
    try{
        throw Exceptions::CEx();
        throw Exceptions::BEx();
        throw Exceptions::AEx();
    }catch(Exceptions::BEx& e){    //LINE-1
        std::cout << "BEx" << " ";
        throw;
    }
    catch(Exceptions::AEx& e){     //LINE-2
        std::cout << "BEx" << " ";
    }
    catch(Exceptions::CEx& e){     //LINE-3
        std::cout << "CEx" << ", ";
        throw 10;
    }
}
int main(){
    try{
        evalEx();
    }catch(int& i){                //LINE-4
        std::cout << "int";
    }
    catch(...){                    //LINE-5
        std::cout << "all";
    }
    return 0;
}
```

What will be the output?

a) `CEx int`

b) `BEx all`

c) `BEx BEx all`

d) `AEx CEx int`

Answer: b)
Explanation:
The function `evalEx()` thorws the exception of type `Exceptions::CEx` which will be caught at `LINE-1` (since it is of base class type). This catch block prints `BEx`, and then re-throws the same exception, which will be caught at `LINE-5` in `main`. Therefore it prints `all`.
Hence, the correct option is b).

Question 3

Consider the following code segment. *[MCQ, Marks 2]*

```
#include<iostream>

class AEx{ public: virtual void printEx() { std::cout << "AEx" << " "; } };
class BEx : public AEx{ public: void printEx() { std::cout << "BEx" << " "; } };
class CEx : public BEx{ public: void printEx() { std::cout << "CEx" << " "; } };
void evalEx(int i){
    try{
        if(i == 0)
            throw CEx();
        else if(i < 0)
            throw BEx();
        else
            throw AEx();
    }
    catch(BEx& e){                //LINE-1
        e.printEx();
    }
    catch(AEx& e){                //LINE-2
        e.printEx();
    }
    catch(CEx& e){                //LINE-3
        e.printEx();
    }
    catch(...){                   //LINE-4
        std::cout << "all";
    }

}
int main(){
    evalEx(-5);
    evalEx(5);
    evalEx(0);
    return 0;
}
```

What will be the output?

a) `BEx AEx CEx`

b) `BEx AEx AEx`

c) `AEx AEx AEx`

d) `BEx BEx BEx`

Answer: a)
Explanation:
Since `printEx` is a virtual function due to dynamic binding in all the catch blocks, `printEx` will bind to its exact derive type. Therefore, the correct option is a).

Question 4

Consider the code segment given below. *[MCQ, Marks 2]*

```
#include <iostream>

void evalEx(int i) {
    i == 0 ? throw "zero" : throw i;
}
int main() {
    try {
        // statement-1
    }
    catch (int& e) {
        std::cout << "int" << " ";
    }
    catch (float& e){
        std::cout << "float" << " ";
    }
    catch (double& e){
        std::cout << "duoble" << " ";
    }
    catch (const char* e) {
        std::cout << "cstring" << " ";
    }
    catch (...) {
        std::cout << "unknown" << " ";
    }
    return 0;
}
```

What will be the outputs in consecutive two runs if `statement-1` is replaced by (i) `evalEx(8.5);` and (ii)`evalEx(0);` respectively?

a) (i) `int` and (ii) `cstring`

b) (i) `double` and (ii) `cstring`

c) (i) `float` and (ii) `unknown`

d) (i) `double` and (ii) `unknown`

Answer: a)
Explanation:
For the call `evalEx(8.5)`, the `double` value is type cast to `int`. Thus, when the exception of `int` type is forwarded to the `main`, it would be caught by `catch(int i){ ... }`.
For the call `evalEx(0)`, the exception type is `const char*` type. Thus, when the exception is forwarded to the `main`, it would be caught by `catch(const char* e){ ... }`.

Question 5

Consider the code segment given below. *[MCQ, Marks 2]*

```
#include<iostream>

_______________________________________    //LINE-1
class Mapping{
    private:
        T1 x;
        T2 y;
    public:
        Mapping(T1 x_, T2 y_){
            x = x_;
            y = y_;
        }
        void show(){
            std::cout << x << " -> " << y << std::endl;
        }
};

int main(){
    Mapping<char, double> p0('X', 4.5);
    Mapping<char> p1(65, 66);
    Mapping<> p2(65, 66);
    p0.show();
    p1.show();
    p2.show();
    return 0;
}
```

Fill in the blank at `LINE-1` such that the output of the program is:

```
X -> 4.5
A -> B
65 -> B
```

a) `template<typename T1, typename T2>`

b) `template<typename T1 = int, typename T2 = char>`

c) `template<typename T1 = char, typename T2 = char>`

d) `template<typename T1 = int, typename T2 = int>`

Answer: b)
Explanation:
From the output, it can be concluded that the default type of `T1` is `int` and `T2` is `char`. Thus, option b) is correct.

Question 6

Consider the code segment given below. *[MSQ, Marks 2]*

```
#include<iostream>

template<class T>
T add(const T& a, const T& b) {
    return a + b;
}

int main() {
    std::cout << ________________;    //LINE-1
    return 0;
}
```

Which of the following statement/s used to fill in the blank at `LINE-1` that results in compiler error?

a) `add(10, 20)`

b) `add(10, 20.5)`

c) `add(10.5, 20.5)`

d) `add(10.5f, 20.5)`

Answer: b), d)
Explanation:
In option a), both the parameters are of type `int`, so `T` would be instantiated to `int`.
In option c), both the parameters are of type `double`, so `T` would be instantiated to `double`.
In option b), the first parameter is of type `int` and the second parameter is of type `double`, so instantiation of `T` is ambiguous.
In option d), the first parameter is of type `float` and the second parameter is of type `double`, so instantiation of `T` is ambiguous.

Question 7

Consider the code segment below. *[MCQ, Marks 2]*

```
#include <iostream>

template <class T, int N = 3>
void genericPrint(T arr[]) {
    for (int i = 0; i < N; i++)
        std::cout << arr[i] << " ";
}

int main() {
    int arr[] = { 18, 30, 35, 22 };
    int n = sizeof(arr) / sizeof(arr[0]);
    genericPrint<int, n>(arr, n) << std::endl;    //LINE-1
    return 0;
}
```

What will be the output?

a) `18 30 35`

b) `18 30 35 22`

c) `18 30 35 22 <garbage-value>`

d) `Compiler error at LINE-1`

Answer: d)
Explanation: In a template declaration, any non-type parameter is a constant. Therefore, at `LINE-1`, the value of `n` is not usable in the constant expression.

Question 8

Consider the code segment given below. *[MCQ, Marks 2]*

```
#include <iostream>
#include <algorithm>
#include <string>
#include <vector>

struct cmp {
    bool operator()(std::string s1, std::string s2) {
        return (s1.length() < s2.length()) ;
    }
};

int main(){
    std::vector<std::string> sVec{"deer", "cat", "rabbit", "sheep"};
    std::sort(sVec.begin(), sVec.end(), cmp());
    for(int i = 0; i < sVec.size(); i++)
        std::cout << sVec[i] << " ";
    return 0;
}
```

What will be the output?

a) `cat deer rabbit sheep`

b) `sheep rabbit deer cat`

c) `cat deer sheep rabbit`

d) `rabbit sheep deer cat`

Answer: c)
Explanation: Since the functor sort the strings of a given array in the ascending order of their length, the correct option is c).

Question 9

Consider the following class definition in C++11. *[MCQ, Marks 2]*

```
class Notification{
    public:
        void alert(const char *msg) {
            std::cout << msg << std::endl;
        }
};
```

Identify the appropriate function pointer declaration that can point to the function `alert` belongs to the class `Notification` as `fp = &Notification::alert;`.

a) `typedef void (Notification::*fp) (const char *);`

b) `void *Notification::fp(const char *);`

c) `void (*Notification::fp)(const char *);`

d) `void (Notification::*fp)(const char *);`

Answer: d)
Explanation:
The appropriate syntax to declaration a function pointer to the member function `alert` of class `Notification` is option d).

Programming Questions

Question 1

Consider the following program. Fill in the blanks as per the instructions given below:

- Fill in the blank at `LINE-1` and `LINE-2` with appropriate statements for class template specialization.
- Fill in the blank at `LINE-3` with appropriate initializer list.

The program must satisfy the given test cases. *Marks: 3*

```
#include<iostream>
#include<cstring>
#include<cstdlib>

template<typename T>
class Manipulator{
    T val;
    public:
        Manipulator(T _val = 0) : val(_val) { }
        T deduct(int d){
            T t = val - d;
            return t;
        }
};

_______________________________    //LINE-1
_____________________________  {   //LINE-2
    char* val;
    public:
        Manipulator(const char* _val = 0) : ________________ { }    //LINE-3
        char* deduct(int d){
            char* buf = (char*)malloc(strlen(val) - d + 1);
            int i;
            for(i = 0; i < strlen(val) - d; i++)
                buf[i] = val[i];
            buf[i] = '\0';
            return buf;
        }
};

int main(){
    int a;
    std::cin >> a;;
    Manipulator<float> f = 100.45;
    Manipulator<const char*> s("programming");
    std::cout << f.deduct(a) << ", ";
    std::cout << s.deduct(a);
    return 0;
}
```

Public 1

```
Input: 3
Output: 97.45, programm
```

Public 2

```
Input: 5
Output: 95.45, progra
```

Private

```
Input: 10
Output: 90.45, p
```

Answer:

```
LINE-1:  template<>
LINE-2:  class Manipulator<const char*>
LINE-3:  val(strdup(_val))
```

Explanation:
For specialized class template declaration (for cstring), `LINE-1` should be filled as `template<>`, and `LINE-2` should be filled `class Manipulator<const char*>`.
At `LINE-3`, the initializer for the given constructor can be written as `val(strdup(_val))`.

Question 2

Consider the following program. Fill in the blanks as per the instructions given below.

- Fill in the blank at `LINE-1` with appropriate template declaration for class `DataSet`.
- Fill in the blank at `LINE-2` with appropriate declaration of array `arr`.
- Fill in the blank at `LINE-3` with appropriate parameter / parameters for function `operator=`.

such that it will satisfy the given test cases. *Marks: 3*

```
#include <iostream>

_______________________________     // LINE-1
class DataSet {
    private:
        _______________;            // LINE-2
        int i;
    public:
        DataSet() : i(-1) { }
        void operator=(__________){ // LINE-3
            arr[++i] = data;
        }
        void print() {
            for (int j = N - 1; j >= 0; j--)
                std::cout << arr[j] << " ";
        }
};
int main() {
    const int n = 3;
    DataSet<char, n> ds1;
    for (int i = 0; i < n; i++) {
        char j;
        std::cin >> j;
        ds1 = j;
    }
    DataSet<int, n> ds2;
    for (int i = 0; i < n; i++) {
        int j;
        std::cin >> j;
        ds2 = j;
    }
    ds1.print();
    ds2.print();
    return 0;
}
```

Public 1

```
Input:
a b c
1 2 3
Output: c b a 3 2 1
```

Public 2

```
Input:
x y z
10 20 30
Output: z y x 30 20 10
```

Private

```
Input:
p q r
6 5 4
Output: r q p 4 5 6
```

Answer:

```
LINE-1:  template<typename T, int N>
LINE-2:  T arr[N]
LINE-3:  T& data
```

Explanation:

We have to declare a generic type in `LINE-1` which will be used to declare an array `arr`. The generic type will be `template<typename T, int N>`. We will declare the array arr with that generic type at `LINE-2` as `T arr[N]`. The parameter to be passed in operator function will be `T& data`.

Question 3

Consider the following program. Fill in the blanks as per the instructions given below:

- Fill in the blank at `LINE-1` with appropriate constructor for structure `Stat`.
- Fill in the blank at `LINE-2` with appropriate header declaration for functor.
- Fill in the blank at `LINE-3` with appropriate return statement.

The program must satisfy the given test cases. *Marks: 3*

```
#include <iostream>

struct Stat {
    int s;
    ____________________________________     //LINE-1
    ___________________________________ {    //LINE-2
        for(int i = 0; i < n; i++)
            s += arr[i];
        double a = (double)s / n;
        ____________________;                //LINE-3
    }
};

int main(){
    int a, b, c[10];
    std::cin >> a;
    for(int i = 0; i < a; i++){
        std::cin >> b;
        c[i] = b;
    }
    int sum = 0;
    Stat st(sum);
    double avg = st(c, a);
    std::cout << st.s << " " << avg;
    return 0;
}
```

Public 1

```
Input: 4 10 20 30 40
Output: 100 25
```

Public 2

```
Input: 6 1 2 3 4 5 6
Output: 21 3.5
```

Private

```
Input: 5 10 -5 6 -9 1
Output: 3 0.6
```

Answer:
`LINE-1: Stat(int& _s) : s(_s) { }`
or
`LINE-1: Stat(int _s) : s(_s) { }`
`LINE-2: double operator()(int arr[], int n)`
`LINE-3: return a`
Explanation:
At `LINE-1`, the parameterized constructor with one parameter can be defined as:
`Stat(int& _s) : s(_s) { }`
or
`Stat(int _s) : s(_s) { }`
Please note that any other variable can also be used a formal parameter to the constructor.
At `LINE-2`, for defining the function header is defined as:
double operator()(int arr[], int n)
At `LINE-3`, the return statement can be written as:
`return a`