Assignment 8

The due date for submitting this assignment has passed. **Due on 2020-03-25, 23:59 IST.**
As per our records you have not submitted this assignment.
1) Consider the program below.

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

namespace expc {
    class Trouble {};
    class Small : public Trouble {};
    class Big : public Trouble {};
};

int main() {
    try {
        using namespace expc;
        throw Big();
    }
    catch (expc::Trouble&) {
        cout << "caught Trouble" << endl;
    }
    catch (expc::Small&) {
        cout << "caught Small Trouble" << endl;
    }
    catch (expc::Big&) {
        cout << "caught Big Trouble" << endl;
    }
    catch (...) {
        cout << "default" << endl;
    }
    cout << "end of program"; // LINE-1
    return 0;
}
```

What will be the output?

a) caught Trouble
   end of program

b) caught Big Trouble
   end of program

c) default end of program

d) caught Trouble

No, the answer is incorrect.
Score: 0
Accepted Answers:

a) caught Trouble
   end of program

2)
Consider the program below.

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

class excp {
public:
    excp(int i) {}
};

int main() {
    try {
        throw excp(15);
    }
    catch (...) {
        cout << "default";
    }
    catch (excp* e) {
        cout << "caught excp" << endl;
    }
    catch (excp e) {
        cout << "caught excp" << endl;
    }
    catch (int e) {
        cout << "caught int" << endl;
    }

    return 0;
}
```

What will be the output/error?

- a) caught excp
- b) default
- c) caught int
- d) Error: catch(...) handler must be the last handler for the try block

No, the answer is incorrect.
Score: 0
Accepted Answers:
- d) Error: catch(...) handler must be the last handler for the try block
3) Consider the following program.

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

void test(int x) {
    try {
        (x < 0) ? throw "negative number " : throw x;
    }
    catch (int e) {
        cout << "int caught ";
        throw;
    }
    catch (char* e) {
        cout << "string caught ";
    }
}

int main() {
    try {
        test(35);
        test(0);
    }
    catch (...) {
        cout << "default " << endl;
    }

    return 0;
}
```

What will be the output?

- a) int caught
- b) int caught default string caught
- c) int caught default
- d) default string caught

No, the answer is incorrect.
Score: 0
Accepted Answers:
- c) int caught default

4)
Consider the following program.

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

int main() {
    try {
        try {
            // Statement-1
        }
        catch (int) {
            cout << "int " << endl;
        }
        catch (char) {
            cout << "char " << endl;
        }
        catch (float) {
            cout << "float " << endl;
        }
        catch (...) {
            cout << "all " << endl;
        }
    }
    return 0;
}
```

What will be the output if (i) statement-1 is throw 10; (ii) statement-1 is throw 'a'; and (iii) statement-1 is throw 3.14;?

- a) (i) int (ii) char, (iii) float
- b) (i) int all (ii) char, (iii) float
- c) (i) int all (ii) char, (iii) all
- d) (i) int, (ii) char, (iii) all

No, the answer is incorrect.
Score: 0
Accepted Answers:
- d) (i) int, (ii) char, (iii) all
5) Consider the code below.

```cpp
template <typename T>
T Max(T x, T y) {
    cout << (x > y ? x : y);
}
```

What shall be the output/error when it is called as:

(i) `Max(10, 20);`
(ii) `Max('A', 'a');`
(iii) `Max(3, 5.0);`

- a) Error: For all the calls, type is not instantiated
- b) (i) 20, (ii) a, (iii) 5.0
- c) (i) 20, (ii) a, (iii) 5
- d) (i) 20, (ii) a, (iii) error: as no matching for `Max(int, double)`

No, the answer is incorrect.
Score: 0
Accepted Answers:
  d) (i) 20, (ii) a, (iii) error: as no matching for `Max(int, double)`
6)

Consider the following program:

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

template<class T, class U = int>
class mydata {
  T n1;
  U n2;
public:
  mydata(T n1, U n2) : n1(n1), n2(n2) {}  
  void show() {
    cout << n1 << " " << n2 << endl;
  }
};

int main() {
  mydata<char, int> d1('a', 10);
  d1.show();

  mydata<char> d2('a', 10);
  d2.show();

  mydata<char> d3('a', 'A');
  d3.show();

  return 0;
}
```

What will be the output/error?

a) a 10
b) a 10
   compiler error: ambiguous type
   a 65
c) a 10
   compiler error: ambiguous type
   a 'A'
d) a 10
   a 10
   a 'A'

No, the answer is incorrect.
Score: 0
Accepted Answers:
a) a 10
   a 10
   a 65
Consider the program below?

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

______________ // LINE-1
class mytype {
    T a, b;
public:
    mytype(T _a, T _b) : a(_a), b(_b) {} 
    _______________ { // LINE-2
        return mytype(a + x.a, b + x.b);
    }

    void show() {
        cout << a << "", " << b << endl;
    }
};

int main() {
    mytype<int> obj1(10, 20);
    mytype<int> obj2(30, 40);
    mytype<int> obj3 = obj1 + obj2;
    obj3.show();

    mytype<double> obj4(3.4, 4.5);
    mytype<double> obj5(10.8, 3.9);
    mytype<double> obj6 = obj4 + obj5;
    obj6.show();

    return 0;
}
```

Fill in the blank at LINE-1 and LINE-2 with appropriate option such that output is:
40, 60
14.2, 8.4

- a) LINE-1: `template<class T>, LINE-2: operator+(mytype x)`
- b) LINE-1: `template<class T = int>, LINE-2: mytype operator+(const mytype& x)`
- c) LINE-1: `template<typename T = int>, LINE-2: mytype operator+(mytype x)`
- d) LINE-1: `template<class T>, LINE-2: mytype& operator+(mytype x)`

No, the answer is incorrect.
Score: 0

Accepted Answers:
- b) LINE-1: `template<class T = int>, LINE-2: mytype operator+(const mytype& x)`
- c) LINE-1: `template<typename T = int>, LINE-2: mytype operator+(mytype x)`
Consider the program below.

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

class Test {
public:
    Test() { cout << "Test::Test() " << endl; }
    ~Test() { cout << "Test::~Test() " << endl; }
};

int main() {
    try {
        Test t;
        throw "error";
    }
    catch (char i) { cout << "caught char " << i << endl; }
    catch (double i) { cout << "caught double " << i << endl; }
    catch (Test&) { cout << "caught Test " << endl; }
    catch (...) { cout << "default " << endl; }

    return 0;
}
```

What will be the output?

- a) Test::Test() default
- b) Test::Test() caught Test Test::~Test()
- c) Test::Test() Test::~Test() default
- d) Test::Test() default Test::~Test()

No, the answer is incorrect.

Score: 0

Accepted Answers:
- c) Test::Test()
  Test::~Test() default