Question 1

Consider the following code segment.

```c++
void myFunction(int test) {
    try {
        if (test)
            throw test;
        else
            throw "Value is zero";
    }
    catch (int i) {
        cout << "CaughtOne ";
    }
    catch (const char *str) {
        cout << "CaughtString ";
    }
}
```

What will be displayed for the following function calls in main().

- `myFunction(-3);` 
- `myFunction(0);`

a) CaughtOne Value is zero  
b) CaughtOne CaughtString  
c) CaughtOne CaughtOne  
d) CaughtString Value is zero

**Answer:** b)  
**Explanation:** A non-zero integer value is considered as true in C++. Hence, for the function call `myFunction(-3)`, *CaughtOne* will be displayed. *CaughtString* will be displayed for `myFunction(0)`. For a detailed explanation, visit the course video lectures for exception handling in C++.

Question 2

Consider the following Template.
template <class T>
T Comp(T x, T y) {
    return x + y;
}

If a = 3.1, b = 5, c = 2, d = 3.7 and if a, b, iC are declared as int and c, d, dC as double. What will be stored in iC and dC for the following calls iC = Comp(a, b); dC = Comp(c, d); in main().

a) 8 and 5
b) 8.1 and 5
c) 8.1 and 5.7
d) 8 and 5.7

Answer: d)
Explanation: a is declared as int hence 3.1 will gets converted to 3 implicitly. Further, iC is declared as int, the result will be integer (8). The result will be double (5.7) for the call dC = Comp(c, d) since all variables are declared as double.

Question 3
Consider the following code segment. Assume that the sizeof(double)= 8. What will be the size of the object derived?

Mark 2

class base {
    double arr[5];
};

class base1 : public base {  };

class base2: public base {  };

class derived: public base1, public base2 {};

a) 40
b) 20
c) 80
d) 88

Answer: c)
Explanation: Class derived ISA a base1 and base2 and hence inherits all its data and function members. But, memory will be allocated only for data members. Hence, the total memory will be 80 (5 (array elements) * 8 (double size) + 5 (array elements) * 8 (double size))
**Question 4**

Consider the following Class Template.  

```cpp
template<class T=int, class U=int>
class Test {
    T x; U y;
public:
    Test(T t, U u): x(t), y(u) { }
    void display() { cout << x << "," << y << endl; }
};
```

What will be the output for the following instantiations in main() ?

Test<char, char>b('a', 'b'); Test<>c('a', 12.9);

a) a,b
   a,12.9  

b) a,b
   97,12  

c) a,b
   a,12  

d) 97,b
   97,12  

**Answer:** b)  

**Explanation:** Default arguments to a class template. Since by default, arguments are `int` to the template. When we make a statement `Test<>c('a', 12.9);` without any explicit data types within `< >`, all arguments will be converted to `int` type.

**Question 5**

What will be the output of the following program?  

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

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

int main() {
    try {
        Test t1;
        throw 98;
    }
    catch(char i) {
```
cout << "Caught Char " << i << endl;
}
catch(double i) {
    cout << "Caught Double " << i << endl;
}
catch(...) {
    cout << "Default" << endl;
}
return 0;
}

a) Created
   Destroyed
   Caught Char b
   Caught Double 98

b) Created
   Caught Char b
   Caught Double 98
c) Created
   Caught Double 98
d) Created
   Destroyed
   Default

**Answer:** d)

**Explanation:** Object instantiation calls the constructor and then calls the destructor when
the scope ends. Since 98 is neither char nor double, catch(...) will be called. Refer course
materials for more details on exception handling.

**Programming Assignment**

**Question 1**

Fill up the missing code segment by following the comments below such that output matches
the test cases.

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

/* Write the function header and body of display() which takes single generic parameter
   and prints it's value */

------------------------- // Provide suitable template signature
void display(____ x) { // Make x as a generic parameter
    cout << x << " ";
}  
```
/* Write overladed display() which takes two generic parameters and prints the values */

----------------- // Provide suitable template signature

void display(____ x, ____ y) { // Make x and y as generic parameters
    cout << x << " " << y << endl;
}

int main() {

double d;
int i;
char c;

cin >> i;
cin >> d;
cin >> c;

display(c);
display(i, d);
display(c, d);

    return 0;
}

Answer:

template <class T>
void display(T x) {
    cout << x << " ";
}

template <class T1, class T2>
void display(T1 x, T2 y) {
    cout << x << " " << y << endl;
}

Explanation: Concept of overloading. Here, we are overloading the function display() with generic parameters.

a. Input:

    10 12.2 c

    Output:
    c
    10 12.2
    c 12.2

b. Input:
8 8 a

Output:
a
8 8
a 8

c. Input:

2 10.8 g

Output:
g
2 10.8
g 10.8

Question 2

Consider the following code. Write the correct swap function to match the test cases. Marks 3

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

// ----------------- Write the Swap function here ---------------------
// --------- You cannot write more than one Swap function -------------
// --------------------------------------------------------------------

// -------------------------------

int main() {
    int a, b;
    double s, t;
    string mr, ms;

    cin >> a >> b;
    cin >> s >> t;
    cin >> mr >> ms;

    Swap(a, b);
    Swap(s, t);
    Swap(mr, ms);

    cout << a << " " << b << endl;
    cout << s << " " << t << endl;
    cout << mr << " " << ms;

    return 0;
}
```
Answer:

```cpp
template <typename T>
void Swap(T& x, T& y) {
    T tmp = x;
    x = y;
    y = tmp;
}
```

**Explanation:** Use template to swap generic parameters.

a. Input: 5 2 11.66 3.3 come wel

Output: 2 5

3.3 11.66
wel come

b. Input: 20 30 2.2 3.3 hello world

Output: 30 20

3.3 2.2
world hello

c. Input: 9 15 77.7 88.8 program c++

Output: 15 9

88.8 77.7

c++ program

Question 3

Consider the following code. Write the correct Min function which returns the minimum of two values to match the test cases. 

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

// ----------------- Write the Min() function here ---------------------
// ------ You cannot write more than one Min() function ---------------
// --------------------------------------------------------------------

int main() {
    int a, b, iMin;
    cin >> a >> b;
    double c, d, dMin;
    cin >> c >> d;
```
iMin = Min(a, b);
cout << "Min(" << a << ", " << b << ") = " << iMin << endl;
dMin = Min(c, d);
cout << "Min(" << c << ", " << d << ") = " << dMin << endl;

return 0;
}

Answer:

template<class T>
T Min(T x, T y) {
    return x < y ? x : y;
}

Explanation:  Use template to find the min. of two values that are of two different type.

a. Input:
   2 3
   -2.2 3.3

   Output:
   Min(2, 3) = 2
   Min(-2.2, 3.3) = -2.2

b. Input:
   8 0
   0.0 3.0

   Output:
   Min(8, 0) = 0
   Min(0, 3) = 0

c. Input:
   -2 -10
   -8 8

   Output:
   Min(-2, -10) = -10
   Min(-8, 8) = -8
Question 4

Consider the following code. Insert the code for error handling using exception, in editable section to match the test cases.

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

class myexception : public exception { // Inherit exception with appropriate visibility
  virtual const char* what() const throw() {
    return "DivideByZero";
  }
};

class DivideByZero {
  public:
    int numerator, denominator;
    DivideByZero(int a = 0, int b = 0) : numerator(a), denominator(b) {} 
    int divide(int numerator, int denominator){
      if (denominator == 0) {
        ___________ // Call exception suitably to handle divide by zero error
      }
      return (numerator / denominator);
    }
};

int main() {
  DivideByZero d;
  int a, b;
  cin >> a >> b;

  try {
    d.divide(a, b);
  }
  catch (exception& e) {
    cout << e.what() << endl;
  }

  return 0;
}

Answer

public exception // throw myexception();

Input:
9
20 0
Output: DivideByZero

Input:
3.3 0
Output: DivideByZero

Input:
10 0
Output: DivideByZero