Q.1) Difference between checked and unchecked Exception?

→

Feature	Checked Excep.	Unchecked Excep
Defn	Exceptions that are checked at compile time	Exceptions that are checked at run time.
Inheritance	subclasses of Exception	Subclasses of Runtime exception
Handling Requirement	Must be handled using try-catch or declared using throws	No need to handle or declare explicitly.
Examples.	IO Exception, SQL Exception. File not found exception	Null pointer exception Array index out of bound exception.

Q2) Define an exception. How it is handled?

→ An exception is an unexpected event that occurs during the execution of a program and disrupts the normal flow of instruction.

- It is handled by following way

1) Try block
code that might generate an exception is placed inside the try block.

2) Throw statement
when an error occurs, the program throws an exception using throw keyword.

3) catch block
The catch block catches and handles the exception thrown by try block

* Syntax:

```
try {

      throw exception type
 }
 catch(exception type) {

   }
```

Ex

```
#include <iostream>
using namespace std;
int main () {
    try {
        int a = 10, b = 0
        if (b==0)
          throw "Division by zero!";
       cout <<a/b;
    } catch (const char* msg) {
       cout << "Error:" <<msg <<endl;
    }
    return 0;
}
```

o/p: Division by zero!

Q.3) Explain with example how can a class template be created.

⟶

A class template allows you to create a class that work with any data type. It is useful for writing generic classes like stacks, queues or linked lists that can be store any data (int, float, chat, etc) without rewriting the class for each ~~type~~ type syntax:

```
template <classT>
class className {
   T var;
  public:
   className (T val) {
        var = val;
   }
   void display () {
     Cout <<"value:" << var << "endl;
```

Ex:
```
   #include <iostreem>
   using namespace std;
   template <class T>
   Class Box {
      T value;
    public:!
      Box (T v) {
        value=v;
      }
     void show() {
       cout <<" value: " << value <<endl;
  }
 }
int main() {
    Box<int> intBox (10);
    Box <float> float Box (3.14);
    Box < string > str Box ("Hello");
    intBox. show();
    floatBox show();
    strBox. show();
    return 0;
}
```

o/p :
```
value: 10
value : 3.14
value : Hello.
```

Q.5) What is the difference betwn opening a file with constructor fun and opening a file with open() fun?

→

1) using construction fun:
This means opening a file directly when creating the file stream object.

2) using open() fun:
This means creating the stream object first and then opening the file using the open method

Ex: constructor method.
```
std::ofstream outFile("data.txt");
if (outFile. is_open()) {
    outFile <<"Hello world!";
    outFile. close();
}
```

constructor method	open method.
open file at the time of object creation.	opens file after object is created
Less flexible - you can't reuse the same object for another file	More flexible - you ca open and close diff files using the same object
Same error handling using is_open()	same error handli using is_open()
Good for simple tasks where you open one file	Better for dynamic situation. Ex: opening multip files in loop.

Q.6) What is Standard Template library?
How is it different from the C++ Standard library?

→ The Standard Template Library (STL) is a collection of generic classes and fun in C++ that provides commonly used data structures and algorithms. It is part of C++ stand Library

Main components of STL.
1) Container = store data

2) Algorithm = Perform operations on containers

3) Iterators = Acts as pointers to navigate through container element

4) Functors = objects used like fun, often passed to algorithms

STL	C++ STL
Subset of C++ Standard Library	The complete library that includes STL
Generic data structures and algorithm	Includes STL + other utilities (I/O, String,
Ex: vector, map, sort, find	iostream, string, chrono, thread, STL, etc