1. Start this practical task with watching lecture 3. In this task, you will need to conduct a small research

about every exception presented in the following list:

a) NullReferenceException

NullReferenceException is caused when the user tries to access member fields, function types, or object references that point to 'null' (nothing). The runtime system is responsible for throwing this exception, however if the programmer is to throw the exception themselves , it is important to detail that the null reference was encountered (for example, "object is null"). The exception can be caught preferably through try-catch. It is better to catch NullReferenceExcaption instead of passing it because it prevents the application from crashing abruptly while also notifying the user. This exception should be avoided, and can be done by ensuring that all object references are not null.

b) IndexOutOfRangeException

IndexOutOfRangeException usually involves arrays and is caused when a statement tries to access an element greater than the maximum allowable index, for example if the user is trying to access an element at index 5 when the array only goes up to 3 indices. The runtime system is responsible this exception, however the programmer can throw this exception themselves, and must detail the problem (for example, "Index is out of range"). This exception can also be caught preferably through the try-catch function. It is better to catch IndexOutOfRangeExeptions instead of passing it because it will prevent the program from crashing and can provide feedback to the user (if its programmed to do so). It is best to try avoid this, this can be done by validating arrays and using try-catches.

c) StackOverflowException

StackOverflowException is only thrown in a checked context. It occurs when a number becomes too big to be represented in bytes. This is because an integer takes four bytes in size, more bytes would be required to represent a larger number. The runtime system is charge of throwing this exception, however the programmer can throw an exception themselves. According to Microsoft, it is not possible to try catch this exemption with a catch block and the program will be terminated automatically. Therefore, it is best to pass the exception instead of catching it. According to SyntaxFix, StackOverFlowException can be avoided by "Writing a code to detect and prevent a stack overflow". For example, If an application depends on recursion (calling itself), the use of counters or state conditions can be used to terminate the recursive loop before StackOverFlowException occurs.

d) OutOfMemoryException

OutOfMemoryException occurs when the program runs out of memory to allocate new objects, therefore not being able to continue the execution of the program. The runtime system is in charge of throwing this exception, however, the programmer can also throw an exception (for example: "Out of memory"). Unlike the rest of the exceptions, it is difficult to effectively catch OutOfMemoryException, as the program can't do much about the lack of memory. Therefore it is

best to pass the exception. It is best to avoid this exception by either, setting constraints on a program, preventing it from becoming too large, increasing memory allocation, or creating more efficient programs.

e) DivideByZeroException

DivideByZeroException occurs when the user attempts to divide a number by zero. The runtime system is in charge of throwing the exception, however, the programmer can also throw an exception (For example: Cannot divide by zero). This exception can be caught using try-catch. It is better to catch this exception rather than pass it because it will prevent the program from crashing and can notify the user (if programmed to do so). This can be avoided by avoiding any calculation that involves dividing a number by zero.

f) ArgumentNullException

ArgumentNullException occurs when at least one of the arguments in a program is null but shouldn't be null. It is not thrown by a runtime system, but instead by "an application or a library" according to Rollbar. However, it can also be thrown by the programmer (For example: Argument cannot be null). This exception can be caught using try-catch. It is better to catch this exception instead of passing it as it will prevent the program from crashing. This can be avoided by checking for null return values and validating arguments.

g) ArgumentOutOfRangeException

This exception occurs when an argument's value is outside a valid range. For example, when the user inputs a negative number when only positive numbers are valid. Usually, the runtime system is in charge of throwing the exception, however, the programmer can do so as well (For example: Argument is out of range). This exception can be caught using try-catch, and is best caught instead of passed as it prevents the program from crashing. This can be avoided by validating input.

h) FormatException

This exception occurs when a format of an argument is invalid, or when a format string is not well formed. An example of this is when the user enters a non-integer item when prompted for their age. The runtime system is in charge of throwing the exception, however, the programmer is able to throw the exception as well (For example: Incorrect formatting). This exception is best caught using the try-catch function instead of passed as it prevents the program from crashing. This can be avoided through validation (try-catch).

i) ArgumentException

This exception occurs when one of the arguments provided to a method is not valid. Both the runtime and programmer can throw this exception. This exception is best being caught using the try-

catch function rather than being passed through the system as it will prevent crashes. This can be avoided by ensuring that arguments meet requirements.

j) SystemException

SystemExeption serves as the “base class” for exceptions namespace. This means that it is a parent class for a wide range of multiple exceptions that can occur during a program execution. The runtime system throws the specific exceptions that make up SystemException. Meaning that the programmer cannot try-catch SystemException as a whole, but can try-catch the individual exceptions in SystemException. It is best to try and catch all of the exceptions as a multitude of them would crash the program. SystemException can be avoided by following the appropriate programming practices and validating everything.