# c++ and Errors

Complete the following two tasks for each of the following code snippets:

1. Circle the line(s) that cause an error.
2. Categorize each of the following code snippets by the type of error that they produce: runtime, compiletime, or no error.
3. You may assume all needed libraries have been #included.

|  |  |
| --- | --- |
| int  } | main() {  int a = 10; std : : string b = "cat" ; std : : cout << (a + b) << std : : endl ; |

1.1

2

3

4

5

This is a compile time error because you are mixing the string type and int type.

|  |  |
| --- | --- |
| int  } | main() {  int a = 10; std : : string b = "cat" ; std : : cout << a << b << std : : endl ; |

2.1

2

3

4

5

No error here

|  |
| --- |
| void PrintContents ( std : : vector<int> v) {  for ( int i = 0; i <= v . size () ; i++) {  std : : cout << v [ i ] << std : : endl ;  }  }  int main( int argc , char∗ argv [ ] ) { std : : vector<int> v = {1 , 2 , 4};  PrintContents (v) ; } |

3.1

2

3

4

5

6

7

8

9

10

Error here: i <= v . size ()

It is a run time error

|  |
| --- |
| struct Book { std : : string t i t l e ;  };  void PrintContents ( std : : vector<Book> v) { for ( int i = 0; i < v . size () ; i++) { std : : cout << v [ i ] . t i t l e << std : : endl ;  }  }  int main( int argc , char∗ argv [ ] ) {  Book b;  b. t i t l e = "BFG" ; std : : vector<Book> v = {b};  PrintContents (v) ; } |

4.1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

No error in code

5.1 int main( int argc , char ∗∗argv ) {

1. std : : cout << argv [0] << std : : endl ;
2. std : : cout << argv [1] << std : : endl ;
3. }

There is no error

5b. What type is argv in the above code snippet?

# it is an array of char arrays

# Static type checking

1. When does static type checking happen?

Static type checking means that the type of a specific variable is checked for type correctness at compiletime.

1. What are at least 3 specific benefits of static type checking?

Detect type errors sooner, more readable code, easier to maintain code

# Python and errors

Useful tips for python:

print(var1, var2) is equivalent to cout « var1 « " " « var2 « endl;. range(number) produces a list of integers from 0 to number - 1. In python 3, "/" is float divide and "//" is integer divide.

|  |
| --- |
| def main() : a = 10 b = "cat"  print (a + b) main() |

1.1

2

3

4

5

6

Run time error

|  |
| --- |
| def main() : a = 10 b = "cat"  print (a , b) main() |

2.1

2

3

4

5

6

No error

|  |
| --- |
| def print\_list ( ls ) :  for i in range ( len ( ls ) + 1) : print ( ls [ i ])  def main() :  ls = [1 , 2 , 4] print\_list ( ls ) main() |

3.1

2

3

4

5

6

7

8

9

It is a run time error

|  |  |  |
| --- | --- | --- |
| def print\_list ( ls ) :  for i in range ( len ( ls ) ) : print ( ls [ i ])  def main() :  ls = [ "cat" , 1236 , True , print\_list ( ls ) main() | False , | 0.123] |

4.1

2

3

4

5

6

7

8

9

No error

5.1 import sys

2

1. def main() :
2. print ( sys . argv [0])
3. print ( sys . argv [1])

6

1. main()

No error

|  |
| --- |
| def main() :  for i in range (10) :  print ("Hello , world ! ") main() |

6.1

2

3

4

5

It is an error with the indentation

# add\_to\_values

|  |
| --- |
| def add\_to\_values( ls , v) :  for i in range ( len ( ls ) ) :  ls [ i ] = ls [ i ] + v |

1

2

3

1. Given the above function definition, write down 6 function calls to add\_to\_values, all with the correct number of parameters and that use a list or a string as values for the first parameter. Which of them produce errors? Make sure at least 2 of the function calls produce errors.

﻿add\_to\_values([1, 2, 3], 5)

﻿add\_to\_values([10, 20, 30], -2)

﻿add\_to\_values(['a', 'b', 'c'], 'd')

﻿add\_to\_values('hello', ' world')

﻿add\_to\_values(123, 5)

﻿add\_to\_values([1, 2, 3], [4, 5, 6])

# Dynamic type checking

1. When does dynamic type checking happen?

This checks the validity of type correctness at runtime

1. What are at least 3 specific benefits of dynamic type checking?

It has high flexibility, simplified code, error detection at the runtime