Name:

1 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, compile time, or no error.
3. You may assume all needed libraries have been `#included`.

```
1  int main() {
2      int a = 10;
3      std::string b = "cat";
4      std::cout << (a + b) << std::endl;
5  }
```

compile time error
can't add int to string

```
1  int main() {
2      int a = 10;
3      std::string b = "cat";
4      std::cout << a << b << std::endl;
5  }
```

no error

```
1  void PrintContents(std::vector<int> v) {
2    for (int i = 0; i <= v.size(); i++) {
3      std::cout << v[i] << std::endl;
4    }
5  }
6
7  int main(int argc, char* argv[]) {
8      std::vector<int> v = {1, 2, 4};
9      PrintContents(v);
10 }
```

no error
if it keep accessing next vector element,
vector will keeping creating one

```
1  struct Book {
2      std::string title;
3  };
4
5  void PrintContents(std::vector<Book> v) {
6      for (int i = 0; i < v.size(); i++) {
7          std::cout << v[i].title << std::endl;
8      }
9  }
10
11 int main(int argc, char* argv[]) {
12     Book b;
13     b.title = "BFG";
14     std::vector<Book> v = {b};
15     PrintContents(v);
16 }
```

no error

```
1  int main(int argc, char **argv) {
2    std::cout << argv[0] << std::endl;
3    std::cout << argv[1] << std::endl;
4  }
```

runtime maybe 3

2 Static type checking

1. When does static type checking happen?

compile time

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

read only
a lot of checking
run fast, saving time
intialization

3 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.

1.
```python
def main():
  a = 10
  b = "cat"
  print(a + b)

main()
```
type error

2.
```python
def main():
  a = 10
  b = "cat"
  print(a, b)

main()
```
no error

3.
```python
def print_list(ls):
  for i in range(len(ls) + 1):
    print(ls[i])

def main():
  ls = [1, 2, 4]
  print_list(ls)

main()
```
index error

4.
```python
def print_list(ls):
  for i in range(len(ls)):
    print(ls[i])

def main():
  ls = ["cat", 1236, True, False, 0.123]
  print_list(ls)

main()
```
no error

5.
```python
import sys

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

main()
```
it depends on argument passed

6.
```python
def main():
  for i in range(10):
  print("Hello, world!")

main()
```
identation error

4 add_to_values

```
1 def add_to_values(ls, v):
2   for i in range(len(ls)):
3     ls[i] = ls[i] + v
```

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([3,4,5],True)
add_to_values("sajkfhksdf",2)——error
add_to_values([10,2,3,77,56],[1,2]) —error
add_to_values("hello",2)—error
add_to_values([3,1,5,75,45],"error")—error
add_to_values([5,6,2,3,1,90],2)

5 Dynamic type checking

1. When does dynamic type checking happen?

runtime

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

be more flexible
easy to generate code
less time on compiling