Week 04 Sample Exam

CSE 232 (Introduction to Programming II)

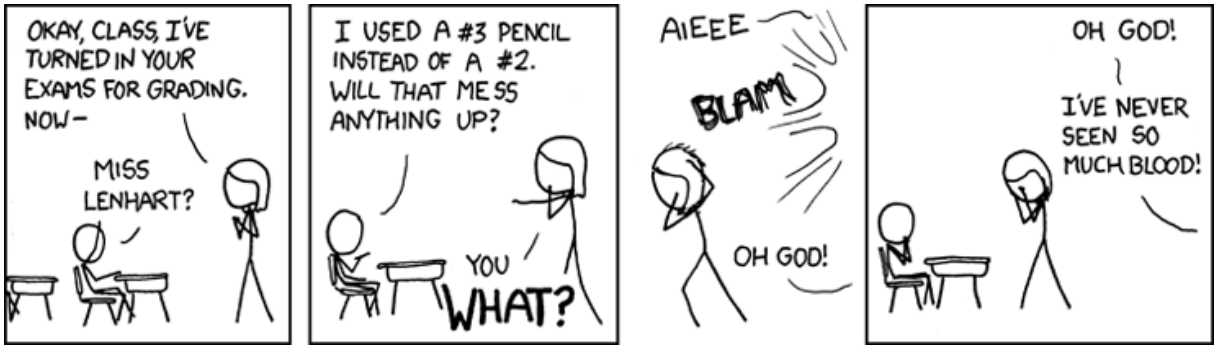
VERSION A

Full Name: .....

Student Number: .....

Instructions:

- DO NOT START/OPEN THE EXAM UNTIL TOLD TO DO SO.
- You may however write and bubble in your name, student number and exam **VERSION/FORM LETTER** (with a #2 pencil) on the front of the printed exam and bubble sheet prior to the exam start. This exam is Version A. Your section doesn't matter and can be ignored.
- Present your MSU ID (or other photo ID) when returning your bubble sheet and printed exam.
- Only choose one option for each question. Please mark the chosen option in both this printed exam and the bubble sheet.
- Assume any needed `#include`s and `using std::...;` namespace declarations are performed for the code samples.
- Every question is worth the same amount of points. There are 55 questions, but you only need 50 questions correct for a perfect score.
- No electronics are allowed to be used or worn during the exam. This means smart-watches, phones and headphones need to be placed away in your bag.
- The exam is open note, meaning that any paper material (notes, slides, prior exams, assignments, books, etc.) are all allowed. Please place all such material on your desk prior to the start of the exam, (so you won't need to rummage in your bag during the exam). Please be sure to bring the required textbook!
- If you have any questions during the exam or when you finish the exam, please raise your hand and a proctor will attend you.



http://xkcd.com/499/ Date Accessed: October 16, 2024

1. What can be done with a pointer after it has been assigned a null value?

   (a) It can be assigned
   (b) It can be initialized
   (c) It can be templated
   (d) It can be aborted
   (e) It can be constructed
   (f) It can be dereferenced

2. What does the following terminal command mean?
   `walk < run`

   (a) `run` will execute, followed by `walk`
   (b) The output of the command will be a `1` if `walk` is less than `run`
   (c) The program `walk` should be executed with the contents of `run` as input
   (d) It will result in an error as logical operators can't be used in the terminal
   (e) `walk` will be executed if its code size is smaller than `run`, otherwise `run` will be executed
   (f) None of the above are true

3. What type does the copy constructor for a class named `Student` return?

   (a) `Student const &`
   (b) `Student *`
   (c) `Student &`
   (d) `Student`
   (e) `Student const *`
   (f) None of the above

4. What advantage is there in returning a reference type (i.e. `string & Func(string & x);`) instead of a regular type (i.e. `string Func(string & x);`)?

   (a) They are identical to each other.
   (b) It avoids an unnecessary copy if the argument.
   (c) It allows the returned object to access the object that is used in the function.
   (d) It allows the function to return a pointer to a dynamically allocated array instead of a statically allocated one.
   (e) None of the above.

5. When are automatic variables (variables that arent dynamically allocated) destroyed?

   (a) When they are deleted.
   (b) When are passed to a function.
   (c) When the loop iteration ends.
   (d) When their copy constructor is called.
   (e) When they are no longer needed.
   (f) When they fall out of scope.

6. Why should you declare a parameter `const` even though the compiler doesnt require it?

   (a) To avoid unnecessary copies.
   (b) To allow the function to be used in a debugger.
   (c) To allow the code to compile faster.
   (d) To allow the compiler to enforce a guarantee made by the developer.
   (e) To enforce runtime assertions.
   (f) All of the above

7. Why do you need to use the first set of parentheses in the following expression (presuming `x` is a pointer to a `vector`)?
   `(*x).clear();`

   (a) Because without them, the compiler would think you are trying to call `clear` on a pointer.
   (b) Because the dot operator has a higher precedence than the dereference operator.
   (c) Because they are needed to avoid a syntax error.
   (d) Because otherwise the behavior would not be the same as `operator->`.
   (e) All of the above.

8. What is the difference between `x` and `y`?
   ```
   for (auto x : vec) ...
   for (auto & y : vec) ...
   ```

   (a) `x` is a copy of each element, where as `y` is a reference.
   (b) If `y` is altered, the element in `vec` changes. Not so for `x`.
   (c) Both of the above are true.
   (d) None of the above are true.

9. Why should you never return the address (a pointer) to a local variable?

   (a) Because the local variable will be destroyed when it goes out of scope (often when the function ends).
   (b) Because the addresses of objects change within functions depending on if there are loops.
   (c) Because doing so makes unnecessary copies of the objects involved.
   (d) Because it leaks memory.
   (e) (a) and (b)
   (f) (b) and (c)
   (g) (c) and (d)
   (h) (b) and (d)

10. What advantage is there in dynamically allocating an array?

    (a) Then the array can hold elements of any size.
    (b) Then the array can be resized with the `std::resize` function.
    (c) Then the size of the array doesn't need to be known at compile time.
    (d) Then the array will not decay into a pointer when passed as an argument.
    (e) Two of (a-d) are correct.
    (f) Three of (a-d) are correct.

11. What is wrong with the following function?

```
int * Add(int * a, int * b) {
  int c = (*a) + (*b);
  return &c;
}
```

    (a) It tries to add the addresses of two pointers.
    (b) It returns the wrong type, it should return a pointer, but instead returns a reference.
    (c) It tries to dereference non-pointer types.
    (d) It returns a pointer to a local variable that will deleted when the function returns.
    (e) Nothing is wrong.

12. What is the the type of x?

```
const auto x = new string[10];
```

    (a) `const string *`
    (b) `string *`
    (c) `string * const`
    (d) `string`
    (e) `const string * const`
    (f) `const string`

13. Redirection allows which of the following operations to be performed?

    (a) The inclusion of libraries that havent been installed yet.
    (b) The ability to use istreams and ostreams to redirect characters from one source to another.
    (c) The ability to search for a file on the file system and add it to the places that the terminal will find when attempting to run executable programs.
    (d) Making a program that writes to standard output instead write to a file.
    (e) The ability to push commits from a local repository to a remote one.
    (f) Changing the address of a pointer to instead point at a different value (especially with respect to streams).

14. When is a dynamically allocated object destroyed?

    (a) When it is assigned to.
    (b) When it is deleted.
    (c) When it falls out of scope.
    (d) None of the above.

15. The pipe character, '|', on the command line indicates what should happen to the commands around it?

    (a) The standard output from the previous command should be redirected to the standard input of the next command
    (b) That output from both commands should be printed to the screen
    (c) The standard input from the previous command should be redirected to the standard output of the next command
    (d) A logical "or" operation should be performed to combine the outputs

16. When should you **NOT** dereference a pointer?

    (a) When its value is null.
    (b) When its value has been deleted
    (c) When its value has been allocated with `new`
    (d) (a) and (b)
    (e) (a) and (c)
    (f) (b) and (c)
    (g) (a), (b), and (c)
    (h) None of the above

17. Why should a function not return a reference to a local variable?

    (a) Because the local variable will then be copied unnecessarily
    (b) Because the variable will be destroyed when the function call returns
    (c) Because the local variables are always dynamically allocated, returning reference to them means the caller is then responsible for deleting them
    (d) (a) and (b)
    (e) (a) and (c)
    (f) (b) and (c)
    (g) (a), (b), and (c)
    (h) None of the above

18. The arrow operator (e.g. `x->y`), is equivalent to which of the following expressions?

    (a) `*x.y`
    (b) `((*x.)y`
    (c) `*(x.y)`
    (d) `(*x).y`
    (e) None of the above

19. What does the [ ] of a `delete` [ ] `x;` indicate?

    (a) That x is subscriptable.
    (b) That x is an array.
    (c) That x is a data structure.
    (d) None of the above.

20. What will change if `struct` is changed to `class` in the following?

```
struct Test {
  double gpa;
 public:
  string name;
 private:
  string comments;
}
```

    (a) All three data members will become private.
    (b) Nothing, it still won't compile because of a missing semicolon.
    (c) The data member `gpa` will be made private.
    (d) The data member `gpa` will be made public.
    (e) All three data members will become public.

21. Which of the following describe dynamically allocating memory of type `long` and of size `size`?

    (a) `new long *lptr = long[size];`
    (b) `long lptr = new *long[size];`
    (c) `new long lptr = *long[size];`
    (d) `long *lptr = new *long[size];`
    (e) `long *lptr = new long[size];`

22. Where are local variables are allocated?

    (a) Stack
    (b) Permanent storage area
    (c) Heap
    (d) Free memory

23. Choose the statement which is **FALSE** with respect to dynamic memory allocation.

    (a) Used for unpredictable memory requirements
    (b) Dynamically allocated memory is automatically deleted when it falls out of scope.
    (c) Memory is allocated in a less structured area of memory, known as heap
    (d) Allocated memory can be changed during the run time of the program based on the requirement of the program
    (e) None of the above

24. The expression `x->y()` is equivalent to what?

    (a) `*((x(.)y))`
    (b) `(*x()).y`
    (c) `(*x).y()`
    (d) `(*x.y)()`
    (e) `*((x.y)())`
    (f) `*(x.y())`

25. Why should read-only parameters be marked as const references?

    (a) It doesn't affect how the program runs, nor its performance, but helps make it clear to the programmer that a variable should not be changed.
    (b) It avoids making a copy and stops the function body from altering the argument.
    (c) It makes changing the argument throw a `std::const_exception`.
    (d) It disallows pointers from being made.
    (e) It allows the function to change the argument within the body, but doesn't affect performance.
    (f) None of the above.

26. Which of the following parameters would copy their argument?
    Example for option (a):
    `void func(string a) {...}`

    (a) `string a`
    (b) `string & b`
    (c) `string const c`
    (d) `string const & d`
    (e) (a) and (b) would both copy their argument.
    (f) (a) and (c) would both copy their argument.
    (g) (c) and (d) would both copy their argument.
    (h) None of the parameters would copy their argument.
    (i) All of the parameters would copy their argument.

27. What does the following Unix instruction do?
    `./a.out < a.txt > b.txt`

    (a) It is equivalent to `(./a.out < a.txt) && (a.txt > b.txt)`.
    (b) It compares the exit code from a.out to the exit code from a.txt (less than) and b.txt (greater than).
    (c) It runs a.out, then runs a.txt, then runs b.txt.
    (d) It compares the sizes of the files of a.out to a.txt (less than) and b.txt (greater than).
    (e) It redirects the contents of a.txt to be the input for a.out and redirects the output from a.out into b.txt.
    (f) None of the above are true.

28. What is implied when a function has a non-const reference for a parameter?

    (a) That the function may read from that argument
    (b) That the function may swap that argument
    (c) That the function may change that argument
    (d) That the function may make a copy of the argument
    (e) None of the above

29. `cat` is a program that prints out a given file to standard output. `sort` is a program that outputs the lines it is given in sorted order. Which of the following commands will output lines of the file "text.txt" in sorted order?

    (a) `cat text.txt > sort`
    (b) `sort << cat text.txt`
    (c) `cat text.txt >> sort`
    (d) `cat text.txt | sort`
    (e) `sort < cat text.txt`
    (f) `sort | cat text.txt`

30. Given the following function, which one of the following claims is true?

```
int Boost(int & in) {
    return ++in * 2;
}
```

    (a) if `x` is 1 and I call `Boost(x)`, `x` will remain as 1
    (b) If `x` is 0 and I call `Boost(x)`, `x` will then be 2
    (c) If `x` is 0 and I call `Boost(x)` it will return 1
    (d) If `x` is 1 and I call `Boost(x)` it will return 4

31. How is the operator (`||`) typically used in C++?

    (a) To test if two boolean expressions are both true
    (b) To output a boolean value to the console
    (c) To link the execution of statements so they occur simultaneously.
    (d) To redirect a value to a file.
    (e) To test if either of two boolean expressions is true

32. How could you redirect the contents of the file `input.txt` to be used as inputs to the executable file `my_exe`, while also redirecting the outputs to the file `output.txt`?

    (a) `input.txt | ./my_exe | output.txt`
    (b) `input.txt > ./my_exe > output.txt`
    (c) `./my_exe < input.txt > output.txt`
    (d) `output.exe < ./my_exe < input.exe`
    (e) `input.txt | ./my_exe > output.txt`

33. Imagine that we have a program named "generate_primes" that writes prime numbers to standard output. And another program named "find_palindromes" that can read numbers from standard input and write to standard output. If we wanted to connect these programs so that the output from "generate_primes" was used as the input to "find_palindromes", which of the following commands would accomplish that?

    (a) `cat generate_primes find_palindromes`
    (b) `generate_primes >> find_palindromes`
    (c) `cp generate_primes find_palindromes`
    (d) `generate_primes | find_palindromes`
    (e) `generate_primes > find_palindromes`

34. If the following program outputs "apple APPLE", what type must Capitalize's first parameter be?

```
string text = "apple"
cout << text << ' ';
Capitalize(text);
cout << text;
```

    (a) string &
    (b) string
    (c) string *
    (d) None of the above allow the behavior demonstrated.

35. What are the uses of non-const reference parameters in functions instead of just using regular pass-by-value?

    (a) Neither of the above are true.
    (b) They can be used to modify the values in the original variables that were passed in as arguments.
    (c) They can be used to "return" additional values to the caller (separate from the return value).
    (d) Both of the above are true.

36. How do you avoid a function argument from being copied (i.e., pass-by-value) in a function call?

    (a) Declare the function parameter as `auto`.
    (b) Declare the function parameter as `const`.
    (c) All of the above
    (d) Declare the function parameter as a reference.

37. What is the output of this program?

```
#include <string>
#include <iostream>
int AddOne(int number) {
        return number + 1;
}
std::string AddOne(std::string const &
str) {
        return str + "1";
}
int main() {
        char c = 'a';
        // ASCII value 97
        std::cout << AddOne(c) <<
std::endl;
}
```

    (a) a1
    (b) b
    (c) 98
    (d) Compilation error, no matching function `AddOne(char)`

38. What will be the output of the following code?

```
void foo(int *p) {
        std::cout << *p << std::endl;
}
int main() {
        int i = 10, *p = &i;
        foo(++p);
}
```

    (a) 11
    (b) Some garbage value or error
    (c) Compile time error

39. What does the `>` character mean in the following terminal command?
    `a.out > abc`

    (a) It compares the exit code of the two files. If a.out returns a larger exit code than abc, then it returns true.
    (b) None of the above
    (c) It compares the contents of the two files. If the contents of a.out are greater alphabetically than abc, then it returns true.
    (d) It does the "greater-than" operation on two variables. If the a.out variable is larger than abd, then it returns true.
    (e) It compares the sizes of the files. If a.out is larger than abc, it returns true.

40. Why should you declare a parameter const even though the compiler doesn't require it?

    (a) To allow the compiler to enforce the promise of constness
    (b) To allow the code to compile faster
    (c) To avoid unnecessary copies
    (d) To enforce runtime assertions

41. In order for file redirection to work, what must a program do?

    (a) It must terminate only after the End-Of-File is encountered
    (b) It must be executed by an IDE
    (c) It must have a using directive for the `std` namespace
    (d) It must conform to the language standard and the style guide
    (e) It must be an executable C++ program
    (f) It must read and/or write to the standard file streams
    (g) It must be compiled with the `-r` flag

42. Why shouldn't you return a reference to a local variable?

    (a) Because local variables will be destroyed when they go out of scope.
    (b) Because references can only be initialized, not declared.
    (c) Wrong, you should return references to local variables.
    (d) Because it violates the property of "Independence"
    (e) Because a function's return type can't have type modifiers.
    (f) Because making a reference will cause an unnecessary copy to be created.
    (g) Because local variables must be const and references could alter them.

43. What is wrong with the following function?

```
bool Func(std::string s) {
  while (!s.empty()) {
    if (s.at(0) == 'a') {
      return true;
    }
    s = s.substr(1);
  }
}
```

    (a) Strings don't have a substr member function
    (b) Control can reach the end of a non-void function
    (c) `!` isn't an operator in C++.
    (d) A string can't be used in a conditional expression
    (e) Nothing is wrong with the function

44. What does the ampersand (`&`) character allow the last two function definitions in section 1.8 (on page 16 of the required textbook) that wouldn't be possible if it was absent?

    (a) It allows the functions to access the argument without copying it.
    (b) It allows the functions to change the argument that they were called with.
    (c) It allows the functions to access member functions of the `vector` class.
    (d) (a) and (b) are both correct.
    (e) (b) and (c) are both correct.
    (f) (a) and (c) are both correct.
    (g) All three statements are true
    (h) None of the statements are true

45. If a function needs read-only access to a large argument like a long string, what type should it usually have? Example provided below:

```
void print(??? x) {
  cout << x;
}
```

    (a) A pointer type, like `string *`
    (b) A const reference type, like `string const &`
    (c) A reference type, like `string &`
    (d) A const pointer type, like `string * const`
    (e) A pointer to const type, like `string const *`
    (f) A const pointer to const type, like `string const * const`
    (g) A regular type, like `string`

46. What is wrong with the following code?

```
char * c_style_string =
  new char[12]{'a', 'b', '\0'};
\\...
delete c_style_string;
```

    (a) The array is initialized with too few values.
    (b) The type of `c_style_string` is wrong.
    (c) The C-Style string doesn't end with a null character.
    (d) The wrong version of delete was used.
    (e) All of the above are issues with the code above.

47. When does a dynamically allocated object's lifetime end?

    (a) When the function it was allocated in returns.
    (b) When the pointer it was assigned to falls out of scope.
    (c) When it is deleted.
    (d) When the `new` operator is called again.
    (e) Never. The object dies only when the program ends.

48. The `at` member function of `std::string` returns what type given that the comment in the following code is true?

```
string s{"abc"};
s.at(0) = 'A';
\\s is now "Abc"
```

    (a) `char const &`
    (b) `char`
    (c) `char *`
    (d) `char &`
    (e) None of the above

49. Why should functions never return references to local variables?

    (a) Because it causes unnecessary copies to be made for every function call.
    (b) Because local variables are destroyed when the function call ends.
    (c) Because dynamically allocated memory returns a pointer, not a reference.
    (d) Because references can only be used for parameters, not for return types.
    (e) None of the above

50. For a class named `Tree`, what is the return type of its constructor?

    (a) `Tree`
    (b) `Tree const`
    (c) `Tree const &`
    (d) `Tree &`
    (e) None of the above

51. The following expression is equivalent to which of the following options?

```
*x.clear()
```

    (a) `*(x.clear())`
    (b) `(*x).clear()`
    (c) `x->clear()`
    (d) (a) and (b) are both correct.
    (e) (b) and (c) are both correct.
    (f) (a) and (c) are both correct.
    (g) All 4 expressions are equivalent
    (h) None of the expressions are equivalent

52. Which line in the following example will cause a compiler error?

```
#include <string>
#include <iostream>
using std::string;
struct Test {
  double gpa;
 public:
  string name;
 private:
  string comments;
};
int main() {
  Test t;
  t.gpa = 2.5;
  t.name = "Josh";
  std::cout << t.comments;
}
```

    (a) `using std::string;`
    (b) `string name;`
    (c) `};`
    (d) `Test t;`
    (e) `t.gpa = 2.5;`
    (f) `t.name = "Josh";`
    (g) `std::cout << t.comments;`
    (h) Two of the above lines cause compiler errors.
    (i) None, the above code would compile.

53. If a program doesn't output to standard output, which of the following commands will change it to write to standard output?

    (a) `./a.out > cout`
    (b) `./a.out < file`
    (c) `cout < ./a.out`
    (d) `./a.out | std`
    (e) `g++ -Wall -COUT`
    (f) None of the above

This page intentionally left blank.