Fill in the Blank 1 point

An enumerated type is a data type where every possible value is defined as a symbolic constant

True or False 1 point

Defining an enumeration allocates memory; therefore, enums are considered to be global variables.

True [✓ False]

Fill in the Blank 1 point

When initializing a variable, the [auto] keyword can be used in place of the variable type to tell the compiler to infer the variable’s type from the initializer’s type.

Fill in the Blank 1 point

C++ provides the [unary] scope resolution operator (::) to access a global variable when a local variable of the same name is in scope.

When we use operator ______ to get user input and put it into a variable, this is called an "extraction".

- [✓ >>]
- <<
- >
- <

Match the library to its functionality

| | |
|---|---|
| The ___________ library contains objects that perform formatted I/O with stream manipulators | iomanip |
| The ___________ library contains objects that perform basic I/O on standard streams. | iostream |
| The __________ library contains objects that perform user-controlled file processing operations | fstream |
| The __________ library contains objects that perform memory formatting | strstream |

Match the blank to the correct value

| | |
|---|---|
| A ____________ stream manipulator permanently changes stream behavior. | sticky |
| A ____________ stream manipulator only effects the stream for the next value. | non-sticky |

Match the stream manipulator to its description

| | |
|---|---|
| Print integers in hexadecimal | hex |
| Print integers in octal | oct |
| Print integers in decimal | dec |

Given this code snippet and assuming that no other stream manipulators were used previously, what would print?

```
int x = 199;

cout << showbase;

cout << setbase(8) << x << " "

     << setbase(10) << x << " "

     << setbase(16) << x << endl;
```

0307 199 0xc7

Given this code snippet and assuming that no other stream manipulators were used previously, what would print?

```
int x = 199;


cout << setbase(8) << x << " "

     << setbase(10) << x << " "

     << setbase(16) << x << endl;
```

307 199 c7

Given this code snippet, match the output to the statement.

```
double doozy = 1.23456;
```

| | |
|---|---|
| cout << setprecision(4) << doozy; | 1.235 |
| cout << scientific << doozy; | 1.234560e+00 |
| cout << fixed << doozy; | 1.234560 |

How many spaces will print in front of HI?

```
cout.width(12);

cout << "HI" << endl;
```

10

Stream manipulator boolalpha is turned off by using noboolalpha.

Match the description to the stream error flag

| | |
|---|---|
| set to true if the wrong type of data is input | failbit |
| set to true if the operation fails in an unrecoverable manner | badbit |
| set to true after end-of-file is encountered after an attempt to extract data beyond the end of the stream | eofbit |
| set to TRUE if none of the bits eofbit, failbit or badbit is set to true for the stream | goodbit |

Which function is used to clear cin's error state flags?

cin.clear()

Given this code snippet, what prints?

```
string MySnack, YourSnack, HerSnack, HisSnack;

stringstream Basket;

Basket << "Apple Banana Orange Pear";

Basket >> MySnack >> YourSnack >> HerSnack >> HerSnack;

cout << HerSnack;
```

Pear

Given this code snippet, what prints?

```
string MySnack, YourSnack, HerSnack, HisSnack;

stringstream Basket, YellowFruits;

Basket << "Apple Banana Orange Pear";

Basket >> MySnack >> YourSnack >> HerSnack >> HerSnack;

YellowFruits << MySnack << HerSnack;

cout << YellowFruits.str();
```

ApplePear

Given this code snippet, what prints?

```
stringstream BigRiver{"Mississippi"};

BigRiver << "Miss";

cout << BigRiver.str();
```

Mississippi

Given this code snippet, what prints?

```
stringstream BigRiver{"Mississippi"};

BigRiver.clear();

BigRiver << "Miss";

cout << BigRiver.str();
```

Mississippi

Given this code snippet, what prints?

```
stringstream BigRiver{"Mississippi"};

BigRiver.str("Mr");

BigRiver << "Miss";

cout << BigRiver.str();
```

Miss

Open a file for output by creating an ofstream object (calling a constructor).

Ios file mode ate seeks to the end of the file before reading/writing.

Member function of ofstream that returns TRUE if file is open and associated with given stream and FALSE if it is not.

[✓ is_open()]

isopen()

open()

Is_Open()

Given a file stream named Alligator, what is the command to close it?

- [✓ Alligator.close();]
- Alligator.is_close();
- Alligator.fclose();
- fclose(Alligator);
- close(Alligator);

Dynamic memory allocation is done from which part of memory?

- Stack
- External Data Segment
- [✓ Heap]
- Unitialized Data Segment
- Initialized Data Segment
- Code Segment

The new operator returns a [pointer] to the type specified to the right of the new operator.

To destroy memory dynamically allocated with new, use free().

True [✓ False]

Will this code snippet compile?

```
int x;

int *y = &x;

delete y;
```

Yes

No

[✓ Yes but will seg fault when run]

What is [illegible]erm used to describe the situation where your program loses the address of dynamically allocated memory before giving it back to the operating system? 2 words

| | |
|---|---|
| Memory | Leak |

Place in the correct category

vector

map

Possible answers

- unordered associative container
- container adapter
- sequence container
- ordered associative container

Match

| | |
|---|---|
| array | fixed-size collections consisting of data items of the same ty |
| vector | collections consisting of data items of the same type that ca |

Fill in the blanks to complete this code snippet to handle bad input

```
while (cin._________())

{

    cin.________();

    cin.ignore(100,'\n');

    cout << "Bad input - reenter ";

    cin << quantity;

}
```

fail

clear

Match the vector member function to its description

| | |
|---|---|
| push_back() | adds a new element to the end of the vector |
| front() | returns the value stored in the first element of the vector |
| back() | returns the value stored in the last element of the vector |
| pop_back() | removes the last element of the vector |

Given this code snippet, what prints?

```
vector<int>AntMan{4,2,7,8,2,1};

AntMan.erase(AntMan.end() - 3);

AntMan.push_back(3);

AntMan.insert(AntMan.begin() + AntMan.size()/2, 3);

for (auto it : AntMan)

    cout << it;
```

4273213

Given this function...

```
void PhoneNumber(string LineNumber, string Prefix="272", string AreaCode="817")

{

        cout << AreaCode << "-" << Prefix << "-" << LineNumber;

}
```

Match the function call to what would print...

| | |
|---|---|
| PhoneNumber("2011"); | 817-272-2011 |
| PhoneNumber("0687","415"); | 817-415-0687 |
| PhoneNumber("0687","415","268"); | 268-415-0687 |

What is this function's signature

```
int TakeTest(int MyChar, char MyInt, long MyLong)
```

TakeTest + int + char + lon

Function overloading should not be used because using it makes your program more complex.

True [✓ False]

Given these two functions

```
int Print(char Alpha='A')

{

        cout << Alpha;

        return Alpha;

}



void Print(char Alpha, int Numeric=65)

{

        cout << Alpha << "=" << Numeric;

}
```

Would code containing these two function calls compile?

Print('A'); Print('A');

Print('A'); Print('A',65);

Given the argument types provided in calls to your function, C++ automatically generates separate

function | template | specializations

Given this code snippet,

```
template<typename INT>

INT funA(INT v1, INT v2)

{

        INT v3 = v1/v2;

        return v3;

}
```

to handle each type of call appropriately.

what would print when this statement is executed?

```
cout << funA(6.4, 2.2);
```

2.90909

## Exam 1 Review

New types in C++

`bool`

Boolean value which can have either 0 or 1

What would this output?

```
bool apple = -1, orange = 0, banana = 1, lemon = 2;
cout << apple << orange << banana << lemon << endl;
```

```
1011
```

# Exam 1 Review

New types in C++

`string`

stream of characters

What would this output?

```
string Exam1Review = {"Isn't this fun?"};
cout << Exam1Review << endl;
```

```
Isn't this fun?
```

# Uniform Initialization

To be warned against unsafe conversions, use the uniform initialization format

```
int Puppy = 10;
int Puppy{10};

string Kitty = "Cat
string Kitty{"Cat"

int Bunny = 1.2;
int Bunny{1.2};
```

```
UI.cpp: In function `int main()':
UI.cpp:14:15: error: narrowing conversion of
`1.2e+0' from `double' to `int' inside { }
[-Wnarrowing]
 int Bunny{1.2};
```

## Exam 1 Review

vector

sequence of elements you can access by index

```
#include <vector>
```

```
vector<int>MyIntVector(10);
vector<char>MyCharVector{'A','B','C','D','E'};
```

# Exam 1 Review

What does this output?

```
vector<int>MyIntVector(10);
vector<char>MyCharVector{'F','H','E','Y','B','C'};

for (int i = 0; i < MyIntVector.size(); ++i)
   MyIntVector[i] = i;

cout << MyCharVector[5] << MyIntVector[7] << endl;
```

```
C7
```

# Exam 1 Review

Now we understand that vector is an object that has member functions

**`size()`**

**`capacity()`**

`front()`

`back()`

`at(`*n*`)`

`pop_back()`

`erase(`*n*`)`

What is the difference between size() and capacity()?

# Exam 1 Review

## Streams

C++ input and output occurs in streams which are sequences of bytes.

- iostream
  - contains objects that perform basic I/O on standard streams
- iomanip
  - contains objects that perform formatted I/O with stream manipulators
- fstream
  - contains objects that perform user-controlled file processing operations
- strstream
  - contains objects that perform memory formatting

## Exam 1 Review

`cin` and `cout`

member functions of `iostream`

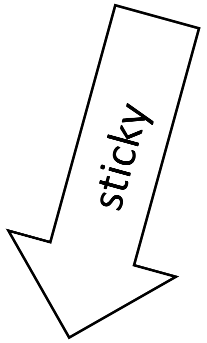
`<<`

stream insertion operator

`>>`

stream extraction operator

# Exam 1 Review



## Stream Manipulators

`dec`, `oct`, `hex`, `showbase` and `setbase`

How to turn off `showbase?` `noshowbase`

`precision, setprecision`

How to turn off `setprecision()?`

Have to reset precision back to orginal value

Default precision length is

# Exam 1 Review

## Stream Manipulators

`scientific`

forces a floating point number to display in scientific notation

`fixed`

forces a floating point number to display a specific number of digits to the right of the decimal

both of these change the stream – how to reset it?

use `defaultfloat` to reset to the default

# Exam 1 Review

## Stream Manipulators

`width`

member function of `cout`

sets the width for the next `cout`

`setw`

stream manipulator

# Exam 1 Review

## Error State Flags

## Member functions of `iostream`

`eof`

- used to determine whether end-of-file has been encountered on the stream
- checks the value of the stream’s `eofbit` data member

`fail`

- used to determine whether a stream operation has failed
- checks the value of the stream’s `failbit` data member

`good`

- used to determine whether a stream operation has failed
- checks the value of the stream’s `goodbit` data member

`bad`

- used to determine whether a stream operation has failed
- checks the value of the stream’s `badbit` data member

`clear`

- used to *restore* a stream’s state to “good” so that I/O may proceed on that stream
- clears `cin` and sets `goodbit` for the stream

# Exam 1 Review

## String Stream Processing

Class `istringstream`

Supports input from a string

Class `ostringstream`

Supports output to a string

member function `str()` returns a copy of the string

Header file __`<sstream>`__ must be included in addition to `<iostream>`

# Exam 1 Review

## File Processing

Class `ifstream`

Supports file input (reading from a file)

Class `ofstream`

Supports file output (writing to a file)

Header file `<fstream>` must be included in addition to `<iostream>`

# Exam 1 Review

## Writing to a File

ios::app

```
ofstream MyOutputFileStream{"outfile.txt", ios::out};

if (MyOutputFileStream.is_open())
{
   MyOutputFileStream << "I am writing this sentence to outfile.txt";
}
else
{
   cout << "The file did not open" << endl;
}

MyOutputFileStream.close();
```

# Exam 1 Review

## Reading from a File

```
_ifstream_ MyInputFileStream{"_makefile_"};
string MyLine;
int LineCounter = 0;

if (MyInputFileStream._is_open()_)
{
   while (getline(_MyInputFileStream_, MyLine))
   {
      cout << "Line " << ++LineCounter << "\t" << MyLine << endl;
   }
}
else
{
   cout << "The file did not open" << endl;
}

MyInputFileStream.close()_;
```

# Exam 1 Review

## `const`

```
#include <iostream>

using namespace std;

int main()
{
  const int x;

  x = 1;

  return 0;
}
```

Would this compile?

No

Why?

Can't declare a variable to be `const` and then try to change it.

Declaring something to be `const` means it is assigned an initial value and is not changed.

# Exam 1 Review

## Passing parameters to/from functions

Pass by value
- Makes a copy
- Safe – function cannot change data
- Overhead of making a copy of the parameter

Pass by reference
- Passes address
- Not as safe – function can change data
- No overhead of making a copy

# Exam 1 Review

Default is pass by value

```
void setEggPlantColor (const std::string& Color)
{
   EggPlantColor = Color;
}
```

How would we make this pass by reference?

How would be make it a safe pass by reference?

## Exam 2 Review
## `namespace`

C++ uses namespace to resolve scope resolution issues

Member's name must be qualified with the namespace name and the scope resolution operator (::)

```
      MyNameSpace::member
```

using directive must appear before the name is used in the program

```
      using namespace MyNameSpace;
```

`using namespace` should not be placed in header files

# Exam 2 Review
# Command Line Parameters

Command line parameters must be separated by any form of whitespace.

If a program was run as

```
./Prog.e abc.txt MyFile zyx
```

The value of `argc` would be 4

Would this line of code compile?

```
      int main(int argv, char *argc[])
      int main(int frog, char *toad[])
```

# Exam 2 Review
# Default Function Arguments

```
int boxVol(int length, int width, int height)
int boxVol(int length=1, int width=1, int height=1)
{
     return length * width * height;
}
```

Add to function definition or to function prototype but not both.

```
int boxVol(int=1, int=1, int=1);
```

## Exam 2 Review
## Unary Scope Resolution Operator

C++ provides the unary scope resolution operator (::) to access a global variable when a local variable of the same name is in scope.

The unary scope resolution operator (::) cannot be used to access a local variable of the same name in an outer block.

A global variable can be accessed directly without the unary scope resolution operator if the name of the global variable is not the same as that of a local variable in scope.

# Exam 2 Review
# Function Overloading

The C++ compiler selects the proper function to call by examining the number, types and order of the arguments in the call.

The combination of a function's name and its parameters types and the order of them is called a signature.

```
int funA(int x, char y);
void funA(int x, char y);
long funA(char y, int x);
```

## Exam 2 Review
## Function Templates

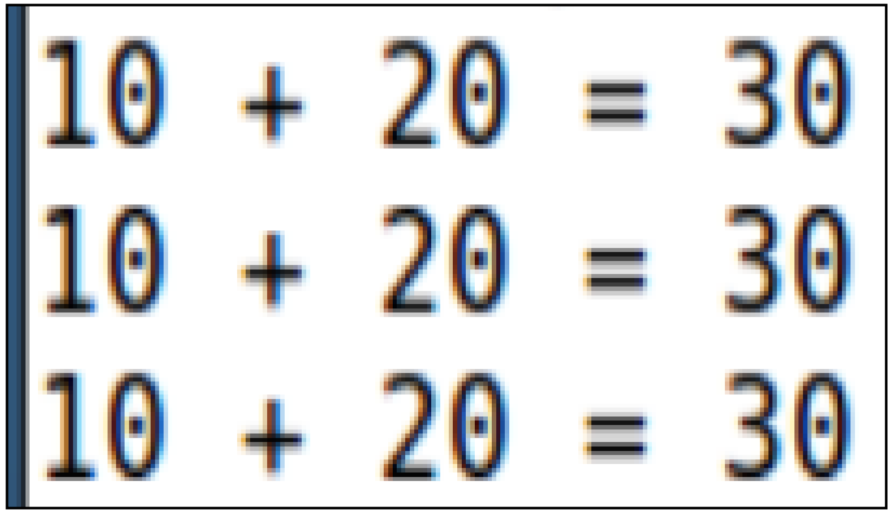
If the program logic and operations are identical for each data type of a function, the overloading may be performed more compactly and conveniently by using function templates.

You write a single function template definition.

Given the argument types provided in calls to the function, C++ automatically generates separate function template specialization to handle each type of call appropriately.

# Exam 2 Review
# Function Templates

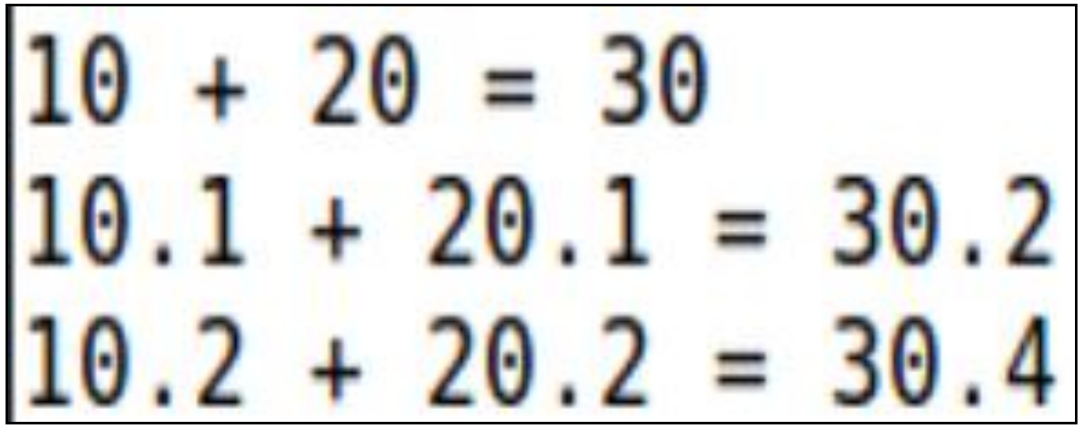
```
void addNumbers(int a, int b)
{
   cout << a << " + " << b << " = " << a + b << endl;
}

int    a{10},   b{20};
float  c{10.1}, d{20.1};
double e{10.2}, f{20.2};

addNumbers(a, b);
addNumbers(c, d);
addNumbers(e, f);
```



```
10 + 20 = 30
10 + 20 = 30
10 + 20 = 30
```

# Exam 2 Review
# Function Templates

```
template <typename T>
void addNumbers( T  a,  T  b)
{
   cout << a << " + " << b << " = " << a + b << endl;
}

int    a{10},   b{20};
float  c{10.1}, d{20.1};
double e{10.2}, f{20.2};

addNumbers(a, b);
addNumbers(c, d);
addNumbers(e, f);
```



```
10 + 20 = 30
10.1 + 20.1 = 30.2
10.2 + 20.2 = 30.4
```

# `makefile`

`make` is UNIX utility that is designed to start execution of a `makefile`

A `makefile` is a list of shell commands that contains a list of rules.

These rules tell the system what commands you want to be executed. Most times, these rules are commands to compile(or recompile) a series of files.

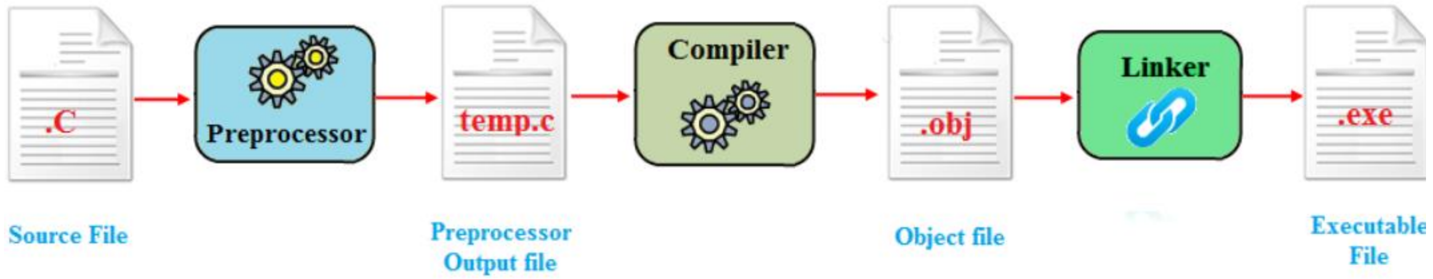
# makefile

```
all : HelloWorld.e

HelloWorld.e : HelloWorld.o
	g++ -g -std=c++11 HelloWorld.o -o HelloWorld.e

HelloWorld.o : HelloWorld.cpp
	g++ -c -g -std=c++11 HelloWorld.cpp -o HelloWorld.o
```

With this explicit `makefile`, calling just “`make`” causes execution to start at rule `all`

Calling “`make HelloWorld.e`” causes execution to start at rule `HelloWorld.e`

Calling “`make HelloWorld.o`” causes execution to start at rule `HelloWorld.o`



compiler

creates an object file

linker

takes in object files and produces an executable file

```
SRC1 = Code2_1000074079.cpp
SRC2 = MyLib.cpp
OBJ1 = $(SRC1:.cpp=.o)
OBJ2 = $(SRC2:.cpp=.o)
EXE = $(SRC1:.cpp=.e)

HFILES = MyLib.h

CFLAGS = -g -std=c++11

all : $(EXE)

$(EXE): $(OBJ1) $(OBJ2)
        gcc $(CFLAGS) $(OBJ1) $(OBJ2) -o $(EXE)

$(OBJ1) : $(SRC1) $(HFILES)
        gcc -c $(CFLAGS) $(SRC1) -o $(OBJ1)

$(OBJ2) : $(SRC2) $(HFILES)
        gcc -c $(CFLAGS) $(SRC2) -o $(OBJ2)
```

## Exam 2 Review
## makefile

```
#makefile for C++ program
SRC1 = ProgramA.cpp
SRC2 = ClassB.cpp
SRC3 = ClassC.cpp
OBJ1 = $(SRC1:.cpp=.o)
OBJ2 = $(SRC2:.cpp=.o)
OBJ3 = $(SRC3:.cpp=.o)
EXE = $(SRC1:.cpp=.e)

HFILES = ClassB.h \
         ClassC.h

CFLAGS = -g -std=c++11
```

```
all : $(EXE)

$(EXE): $(OBJ1) $(OBJ2) $(OBJ3)
	g++ $(CFLAGS) $(OBJ1) $(OBJ2) $(OBJ3) -o $(EXE)

$(OBJ1) : $(SRC1) $(HFILES)
	g++ -c $(CFLAGS) $(SRC1) -o $(OBJ1)

$(OBJ2) : $(SRC2) $(HFILES)
	g++ -c $(CFLAGS) $(SRC2) -o $(OBJ2)

$(OBJ3) : $(SRC3) $(HFILES)
	g++ -c $(CFLAGS) $(SRC3) -o $(OBJ3)
```

# Exam 2 Review
# Range based `for`

## Iterates through entire range of container

```
vector<string> Ranger{"Hello","there","how","are","you","?"};

range_variable-type range_variable : container_name

for (auto it : Ranger)
     cout << it << " ";
```

Won't go out of bounds.

## Exam 2 Review
## `new`

Use the `new` operator to dynamically allocate the exact amount of memory required to hold an object at execution time.

The object created with `new` is created in the heap

The object created without `new` is created in the stack

# Exam 2 Review
## `new`

To destroy a dynamically allocated object, use

```
delete ptr;
```

# Exam 2 Review
## `new`

Not releasing dynamically allocated memory when it's no longer needed can cause the system to run out of memory prematurely. This is sometimes called a "memory leak"

After you delete a block of dynamically allocated memory, be sure not to delete the same block again. One way to guard against this is to immediately set the pointer to `nullptr`.

Deleting a `nullptr` has no effect.  Deleting the same pointer again will cause a core dump.

A stream that is ready to accept or produce data has a stream state of

A) Good

B) Bad

C) Fail

D) Eof

Which command will invoke Makefile rule "main"?

A) make

B) make main

C) make rule main

D) make -r main

The statement "int i = 5;" stores the integer in:

A) Cache Memory

B) Heap Memory

C) Stack Memory

D) None of the above.

The implementation for C++ template Vector should be defined in

A) vector.h

B) vector.cpp

C) vector.tmpl

D) templates.cpp

To stream out the number 3192 as a hex number with leading 0x (i.e., 0xc78), use

A) std::cout << 3192; // hex is the default format

B) std::cout << std::fullhex << 3192;

C) std::cout << std::hex << std::showbase << 3192;

D) std::cout(hex) << 3192;

std::cout is similar to which of these C functions?

A) scanf

B) sprint

C) printf

D) None of the above

Which one of these streams will take input from a file?

A) ofstream

B) stringstream

C) cerr

D) ifstream

## Scoped enumeration

```
enum class Player1Status {CONTINUE, WON, LOST};
enum class Player2Status {LOST, CONTINUE, WON};

Player1Status P1Stat;
Player2Status P2Stat;

P1Stat =
P2Stat =
```