Complete the following sentence (correctly):

If a programmer doesn't define a copy constructor to copy objects, _____.

Your answer is correct

- ○ the compiler uses the default constructor instead
- ○ the compiler shows an error
- ○ the compiler implicitly defines one having no statements
- ◉ the compiler explicitly defines one that does a memberwise copy

Consider the following function template and variable declarations:

```
template <typename KeyType, typename ValueType> ValueType storePair(KeyType k, ValueType v){
    // ...
    return v;
}

std::string s = "hello";
int v = 43;
double d = 84.0;
std::vector<std::string> vec;
```

Which of the following are appropriate invocations of the function?

Your answer is correct

- ☐ int i = storePair(KeyType, ValueType);
- ☑ int i = storePair(s,v);
- ☐ double dd = storePair(d,vec);
- ☑ double dd = storePair(vec,d);
- ☐ double dd = storePair(T,T);

```
#include <iostream>
#include <vector>
#include <cmath>

int main(){
   const int PAGE_SIZE = 4;
   std::vector<int> sizes = {2, 6, 8, 11, 0, 13, 10};
   std::vector<int> v;
   std::cout << v.size() << ' ';
   for ( int i=0; i<sizes.size(); i++ ){
      int pages = std::ceil(static_cast<double>(sizes.at(i)) / PAGE_SIZE);
      v.resize( pages * PAGE_SIZE );
      std::cout << v.size() << ' ';
   }
   return 0;
}
```

What is the output of the above program?

Your answer is correct

0 4 8 8 12 0 16 12

What does the following program print out?

```
#include <iostream>

void numSeries(int num) {
   if (num == 0)
      std::cout << 1 << " ";
   else {
      numSeries( num - 1 );
      num = num + 1;
      std::cout << num*2 << " ";
   }
}

int main() {
   numSeries(2);
   return 0;
}
```

Your answer is correct

1 4 6

The following code is supposed to print the minimum of the values in the vector `userVals`. Which **XXX** and **YYY** correctly output the smallest element?

```
std::vector<int> userVals; // inititalised to have several values
XXX
for (i = 0; i < userVals.size(); ++i) {
   if (YYY) {
      minVal = userVals.at(i);
   }
}
std::cout << "Min: " << minVal << std::endl;
```

Your answer is correct

○ **XXX**: `minVal = 0;`

YYY: `userVal < minVal`

○ **XXX**: `minVal = userVals.at(0);`

YYY: `userVals.at(i) > minVal`

○ **XXX**: `minVal = 0;`

YYY: `userVal > minVal`

◉ **XXX**: `minVal = userVals.at(0);`

YYY: `userVals.at(i) < minVal`

Which of the following expressions generates a random integer in the interval -10...10 ?

Your answer is correct

- ○ rand() % 20
- ○ (rand() % 20) - 10
- ○ rand() % (10 + 1)
- ◉ (rand() % 21) - 10
- ○ rand() % 10

```
char selector (int x){
     char result;
     switch (x){
          case 32:
          case 33:
               result = 'Z';
               break;
          case 34:
              result = 'A';
          case 35:
              result = 'K';
          case 36:
              result = 'M';
              break;
          default:
              result = '@';
              break;
     }
     return result;
}
```

Which of the following statements are true?

Your answer is correct

- [x] selector(31) returns '@'
- [ ] selector(36) returns 'N'
- [x] selector(34) returns 'M'
- [ ] selector(34) returns 'A'
- [x] selector(32) returns 'Z'
- [ ] selector(31) returns 'Z'

What is the error in the following code fragment?

```
ifstream inFS;
int numBooks;
int numStudents = 40;
inFS >> numBooks;
std::cout << "Books per student: ";
std::cout << numBooks / numStudents;
```

Your answer is correct

- (●) The file stream has not been properly opened.
- ( ) The >> operator cannot be used here.
- ( ) The file stream is not large enough.
- ( ) No error.

You intend to shrink a vector `v` by two elements, and by mistake you call `v.resize(-2)`. What happens in that case?

Your answer is correct

- ○ The program will be aborted by an `assert()` in the `resize()` method.
- ◉ `resize()` will throw an exception.
- ○ The program will crash.
- ○ Nothing. The vector will be set to size 0.

```
double d = 1.0;
while ( d > 0.0 ){
    d /= 3.0;
}
```

Consider the above loop. Which of the following statements is true?

Your answer is correct

- ○ After running for a while, the loop will generate a runtime error.
- ○ This loop will run forever.
- ○ This code does not compile.
- ◉ This loop will terminate after a while.

```
#include <iostream>

class SimpleItem {
    public:
        void printNums();
        SimpleItem();
        SimpleItem(int val2);
        SimpleItem(int val1, int val2);
    private:
        int num1, num2;
};

SimpleItem::SimpleItem() : num1(0), num2(0) {};
SimpleItem::SimpleItem(int val2) : num1(-1), num2(val2) {};
SimpleItem::SimpleItem(int val1, int val2){
    num1 = val1; num2 = val2;
}

void SimpleItem::printNums(){
    std::cout << num1 << ' ' << num2 << ' ';
}

int main(){
    SimpleItem* myItem1 = new SimpleItem(10,12);
    SimpleItem myItem2(42);
    SimpleItem* myItem3 = new SimpleItem();

    myItem1->printNums();
    myItem2.printNums();
    (*myItem3).printNums();

    return 0;
}
```

What is the output of this program?

Your answer is correct

10 12 -1 42 0 0

For a given recursive function, which of the following statements are correct?

Your answer is partially correct (See correct answer below)

- ☐ When the recursive invocations always reduce the problem size, a stack overflow will never happen.
- ☐ A recursive function must have exactly one base case that can be computed without a recursive invocation.
- ☐ When the recursive invocations always reduce the problem size, a base case is guaranteed to be reached.
- ▣ Each recursive function invocation must reduce the problem size by some amount.
- ▣ A recursive function must have one or more base cases that can be computed without a recursive invocation.

Which of the following statements are correct?

Your answer is partially correct (See correct answer below)

- ☐ A good C++ compiler can avoid situations in which memory leaks occur.
- ☑ We speak about a memory leak when a pointer to a dynamically allocated region of memory gets out of scope before the memory has been freed.
- ☐ We speak of a memory leak when the runtime stack grows beyond its imits and overwrites other parts of memory.
- ☐ In any case, a function that allocates a memory region must also free the region before it returns.
- ☑ Destructors are meant to avoid memory leaks.

What is the output of the following code?

```
#include <iostream>

int main(){
   for (char letter1 = 'a'; letter1 < 'c'; ++letter1){
      for (char letter2 = '5'; letter2 <= '8'; ++letter2){
         std::cout << letter2 << letter1;
      }
   }
   return 0;
}
```

Your answer is correct

5a6a7a8a5b6b7b8b

What is the final value of *cycles*?

```
int rows = 10, columns = 5, cycles = 12;

if ( rows < 5 ){
     if ( columns > 3 ){
          cycles = 1;
     }else{
          cycles--;
     }
}else{
     if ( columns > 3 ){
          cycles += 6;
     }else{
          cycles = 1;
     }
}
```

Your answer is correct

18

```
#include <iostream>

int main(){
   int *ints = new int[10];
   for ( int i = 0; i < 10; i++ ){
      ints[i] = 10-i;
   }
   int *ptr = &ints[3];
   for ( int i=-3; i<7; i++ ){
      ptr[i] += i;
   }
   for (int i = 0; i<10; i++ ){
      std::cout << ints[i] << ' ';
   }
   return 0;
}
```

What is the output of this program?

Your answer is correct

7777777777

```
class Test{
    public:
        char getM() const { return m; }
        void setM(char newM) { m = newM; }
        Test(char initM) : m(initM) {}
    private:
        char m;
};

Test t('a');
t.m = 'b';
char x = t.getM();
```

Which statements are true about the code above?

Your answer is correct

- [x] This code does not compile.
- [x] If we would have used setM() instead of the assignment, at the end, x would have the value 'b'.
- [ ] At the end, x has the value 'a'.
- [ ] At the end, x has the value 'b'.

Fill in the blank:

```
template<typename BaseType>
_____ max2(BaseType i, BaseType j) {
    return (i>j) ? i : j;
}

template<typename BaseType>
_____ max3(BaseType i, BaseType j, BaseType k) {
    return (i>j) ? max2(i,k) : max2(j,k);
}
```

Your answer is correct

- ☐ double
- ☑ BaseType
- ☐ T
- ☐ int

Which of the following most accurately describes the vector v when the output is 1 (`s = true`)? Assume v is a large vector of integers.

```
int i;
bool s;
for (i = 0; i < v.size(); ++i) {
   if (v.at(i) < 0) {
      s = true;
   }
   else {
      s = false;
   }
}
cout << s;
```

Your answer is correct

- ○ some value other than the last is negative
- ○ first value is negative
- ◉ last value is negative
- ○ all values are negative

The following program generates an error. Why?

```
const int NUM_ELEMENTS = 5;
std::vector<int> userVals(NUM_ELEMENTS);
unsigned int i;
userVals.at(0) = 1;
userVals.at(1) = 7;
userVals.at(2) = 4;

for (i = 0; i <= NUM_ELEMENTS; ++i) {
   cout << userVals.at(i) << endl;
}
```

Your answer is correct

- ○ The vector `userVals` has 5 elements, but only 3 have values assigned.
- ○ Variable `i` is declared as an unsigned integer.
- ◉ The for loop tries to access an index that is out of the vector's valid range.
- ○ The integer `NUM_ELEMENTS` is declared as a constant

```
#include <iostream>

int main(){
  srand( rand() );
  std::cout << rand() << std::endl;
  return 0;
}
```

Which statement is true about the program above?

Your answer is correct

- ○ No matter how often or on which computer the program runs, 42 will never be printed.
- ◉ Each time the program runs, the same number is printed.
- ○ Each time the program runs, 42 is printed.
- ○ Each time the program runs, a different (random) number is printed.

```
int factorial(unsigned int n){
    if ( n == 0 ) return 1;
    return n * factorial(n-1);
}
```

For the above function, which of the following statements are true?

Your answer is partially correct (See correct answer below)

- ☐ The given function runs more efficiently than a loop-based version.
- ☑ The given function directly implements the corresponding mathematical definition.
- ☑ Calling factorial(-1) will most likely cause a stack overflow.
- ☐ For n>1, factorial() always returns n! (according to the mathematical definition)

If the first available heap memory location is at 4200, what is the value of variable `myAge` at the moment when the code prints to `cout`?

```
#include <iostream>
int a = 10;
int main() {
   int b;
   int* myAge = nullptr;
   b = 20;
   myAge = new int;
   std::cout << "Here I am!\n";
   *myAge = 30;
   delete myAge;
   return 0;
}
```

Your answer is correct

- ○ 30
- ○ not allocated
- ○ `nullptr`
- ◉ 4200

In the following code, which variables will have the value 10.5 ?

```
// instead of static_cast<double>(value)
// you can also think of double(value)
//
int a = 17;
int b = 4;
int c = 2;
double d1 = (a+b)/c;
double d2 = static_cast<double>((a+b) / c );
double d3 = (a+static_cast<double>(b))/c;
double d4 = (a+b)/static_cast<double>(c);
```

Your answer is correct

- ☐ d1
- ☐ d2
- ☑ d3
- ☑ d4

Consider the following code:

```
#include <iostream>

int main(){
    int i;
    while ( !std::cin.eof() ){
        std::cin >> i;
        std::cout << i << ' ';
    }
    return 0;
}
```

If the input from cin is "1 2 3 4 " (without the quotation marks), what is the output of the program?

Your answer is correct

1 2 3 4 4

Which is true regarding how functions work?

Your answer is correct

- (●) A function's local variables are discarded upon a function's return; each new call creates new local variables in memory.
- ( ) After a function returns, its local variables keep their values, which serve as their initial values the next time the function is called.
- ( ) A return address indicates the value returned by the function.
- ( ) If a function returns a variable, the function stores the variable's value until the function is called again.

Consider the following code fragment. Which of the given statements are true?

```
class ExceptionType1{};
class ExceptionType2{};
class ExceptionType3{};

try {
   ...
   // If error detected
   throw ExceptionType1();

   ...
   // If error detected
   throw ExecptionType2();

   ...
   // If error detected
   throw ExceptionType3();
}
catch (ExceptionType1& excptObj) {
   // print error message 1
}
catch (ExceptionType2& excptObj) {
   // print error message 2
}
```

Your answer is correct

- ☐ If an object of type ExceptionType3 is thrown, the second catch block will execute.
- ☐ If an object of type ExceptionType1 is thrown, both catch blocks will execute.
- ▣ A second catch block can never execute immediately after a first one executes.
- ▣ If an object of type ExceptionType1 is thrown, only the first catch block will execute.
- ▣ If an object of type ExceptionType3 is thrown, the exception will be handled by the code calling this code fragment.

```
#include <iostream>

int x=1, y=2, z=4;

int bar(int &y){
   int x = 22;
   {
      int x = 33;
      x++;
      y = x;
   }
   return x;
}

int main(){
   int y = 13;
   z = bar(x);
   std::cout << x << ' ' << y << ' ' << z ;
   return 0;
}
```

What is the output of this program?

Your answer is correct

34 13 22

```
bool check(int a, int b){
  if ( ( a<0 ) || ( b<0 ) ) throw std::runtime_error("bad parameters");
  int c = a*b;
  return ( c >= 0 );
}
```

Consider the function above. Which of the following statements are true?

Your answer is correct

- [x] This function can only return false if it does not throw a runtime_error.
- [x] This function returns false on some value combinations for a and b.
- [ ] This function never returns false.
- [ ] This function always returns true.

```
#include <iostream>
#include <stack>

void printTop1(std::stack<int> &st){
  int t = st.top();
  st.pop();
  std::cout << t << ' ';
}

void printTop2(std::stack<int> st){
  int t1 = st.top();
  st.pop();
  int t2 = st.top();
  st.pop();
  std::cout << t1 << ' ' << t2 << ' ';
}

int main(){
  std::stack<int> s;
  s.push(3);
  s.push(2);
  s.push(1);
  printTop2(s);
  printTop1(s);
  printTop1(s);
  return 0;
}
```

What is the output of the above code?

Your answer is correct

1 2 1 2

What is the output for printNum(25)?

```
void printNum(int num) {
   if(num == 0)
      return;
   else{
      printNum(num/2);
      int n = num%2;
      cout << n;
   }
}
```

Your answer is correct

11001

Which of the following statements are correct?

Your answer is correct

- [ ] For a function that may contain a throw, all of the function's statements, including the throw, must be surrounded by a try block.
- [x] If no throw is executed in a try block, then the subsequent catch block is not executed.
- [x] A compiler generates an error message if a try block is not immediately followed by a catch block.
- [ ] A throw executed in a function automatically causes a jump to the last return statement in the function.
- [ ] After an exception is thrown, and a catch block executes, execution resumes after the throw statement.
- [x] A goal of exception handling is to avoid polluting normal code with distracting error-handling code.

Supposed you wrote a function as part of your program that computes the circumference of a circle from its radius *r*: how do you **appropriately** prepare your function for the case of the parameter *r* being negative?

Your answer is correct

- ○ I'll throw an exception.
- ◉ I'll add an *assert* statement that checks *r* before the computation.
- ○ I'll ask the user to correct the value by printing a statement to *cout*.
- ○ I do nothing at all: garbage in, garbage out.

```
class Rectangle {
    public:
        void setLength ( double fullLength );
        void setWidth (double fullWidth ) {
            width = fullWidth;
        }
    private:
        double length, width;
};

void Rectangle::setLength ( double fullLength ){
    length = fullLength;
}
```

Which of the following statements are true?

Your answer is correct

- [x] Inside the class definition, setLength is declared, but not defined.
- [ ] Inside the class definition, setWidth is declared, but not defined.
- [ ] Inside the class definition, setLength is defined, but not declared.
- [x] It would be better class interface design to declare and define both setWidth() and setLength() the same way.
- [ ] setWidth()'s use of width is an error because width is only declared after this use.
- [x] setWidth() is an inline member function.

Which of the following are valid comments in C++? (valid means: the compiler will not complain...)

Your answer is partially correct (See correct answer below)

☐
```
/*
numKids = 2; /* typical number */
numCars = 5;
*/
```

☑
```
// repeat "work" until done //
```

☑
```
/* Determine width and length of the rectangle,
compute the volume of the cube,
and return the square root of the volume. */
```

☑
```
/*
numKids = 2; // typical number
numCars = 5;
*/
```

☑
```
/*
** Author: Donald D.
** Year: 2018
** Copyright (c): Vrije Universiteit
*/
```

☐
```
// Print "hello"
then print "world"
finally, return 0
//
```

☑
```
//
// numKids = 2; /* typical number */
// numCars = 5;
//
```

Identify the error in the recursive function that finds the sum of the digits.

```
int digitSum(int number) {
   int sum = 0;
   if(number >= 0) {
      sum += (number % 10);
      sum += digitSum(number / 10);
      return sum;
   }
   else {
      return sum;
   }
}
```

Your answer is correct

- ○ Incorrect base case condition. The recursive calls straight away go to the base case leading to no recursion.
- ○ There is no base case, and hence the recursive calls never return.
- ◉ Incorrect base case condition. The recursive calls do not reach the base case leading to infinite recursion.
- ○ There is no recursive case, and hence the function is not recursive.