Started on	Thursday, 9 March 2023, 5:04 PM
State	Finished
Completed on	Thursday, 9 March 2023, 10:49 PM
Time taken	5 hours 44 mins
Grade	**60.00** out of 60.00 (**100**%)

Question **1**

Correct

Mark 5.00 out of 5.00

Given the following declarations:

```
enum CurrencyType {
  US_DOLLAR, CAN_DOLLAR, POUND, EURO, YEN, YUAN, RUPEE, MEX_PESO
};

CurrencyType currency;
```

which of these subsequent statements compile?

Select one or more:

- [x] `bool b = MEX_PESO <= YEN;` ✔
- [ ] `currency--;`
- [x] `currency = CAN_DOLLAR;` ✔
- [ ] `currency = MEX_PESO + YEN;`
- [ ] `currency = POUND * 2;`
- [x] `std::cout << currency;` ✔
- [x] `int i = static_cast<int>(currency);` ✔
- [ ] `std::cin >> currency;`
- [ ] `for (currency=US_DOLLAR; currency <= MEX_PESO; ++currency) std::cout << "*";`
- [ ] `++currency;`

The correct answers are: `currency = CAN_DOLLAR;`, `std::cout << currency;`, `int i = static_cast<int>(currency);`, `bool b = MEX_PESO <= YEN;`

Question **2**

Correct

Mark 5.00 out of 5.00

Given the following declarations:

```
enum class CurrencyType {
  US_DOLLAR, CAN_DOLLAR, POUND, EURO, YEN, YUAN, RUPEE, MEX_PESO
};

CurrencyType currency;
```

which of these subsequent statements compile?

Select one or more:

- [ ] `currency = CurrencyType::MEX_PESO + CurrencyType::YEN;`
- [ ] `currency = CurrencyType::POUND * 2;`
- [x] `currency = CurrencyType::CAN_DOLLAR;` ✔
- [x] `bool b = CurrencyType::MEX_PESO <= CurrencyType::YEN;` ✔
- [ ] `std::cout << currency;`
- [ ] `std::cin >> currency;`
- [ ] `currency--;`
- [ ] `++currency;`
- [ ] `for (currency=CurrencyType::US_DOLLAR; currency <= CurrencyType::MEX_PESO; ++currency) std::cout << "*";`
- [x] `int i = static_cast<int>(currency);` ✔

The correct answers are: `currency = CurrencyType::CAN_DOLLAR;`, `int i = static_cast<int>(currency);`, `bool b = CurrencyType::MEX_PESO <= CurrencyType::YEN;`

Question **3**

Correct

Mark 5.00 out of 5.00

Which of the following definitions are *scoped enumerations*? Choose all appropriate definitions.

Select one or more:

- [ ] `enum MyEnum : char { /* some enumeration constants */ };`
- [ ] `enum MyEnum { /* some enumeration constants */ };`
- [x] `enum struct MyEnum : char { /* some enumeration constants */ };` ✔
- [ ] `enum MyEnum { /* some enumeration constants */ };`
- [x] `enum struct MyEnum { /* some enumeration constants */ };` ✔
- [x] `enum class MyEnum { /* some enumeration constants */ };` ✔

Your answer is correct.

The correct answers are: `enum struct MyEnum { /* some enumeration constants */ };`, `enum struct MyEnum : char { /* some enumeration constants */ };`, `enum class MyEnum { /* some enumeration constants */ };`

Question **4**

Correct

Mark 10.00 out of 10.00

To represent some of the punctuators in the English language, define a scoped enumeration type **Punctuation** with enumerators **Period** [to represent period character], **Colon** [to represent colon character], **Semicolon** [to represent semicolon character], **Question** [to represent question mark character], **Exclamation** [to represent exclamation point character], and **Comma** [to represent comma character] with the caveat that every object of type **Punctuation** must occupy one byte of memory.

Next, write a function **pun_to_str** that satisfies the following use cases:

```
std::cout << pun_to_str(Punctuation::Period);      // prints . character to standard output
std::cout << pun_to_str(Punctuation::Colon);       // prints : character to standard output
std::cout << pun_to_str(Punctuation::Semicolon);   // prints ; character to standard output
std::cout << pun_to_str(Punctuation::Question);    // prints ? character to standard output
std::cout << pun_to_str(Punctuation::Exclamation); // prints ! character to standard output
std::cout << pun_to_str(Punctuation::Comma);       // prints , character to standard output
```

C++ standard library header **<string>** is included. Adding other standard library headers will prevent your code from compiling!!!

```
4  // Standard library header <string> is included. You cannot include any other header files!!!
5  // Define function pun_to_str ...
6
7  enum class Punctuation : unsigned char {
8   Period = '.', Colon = ':', Semicolon = ';',
9   Question = '?', Exclamation = '!', Comma = ','
10 };
11 std::string pun_to_str(Punctuation p) {
12  char c = static_cast<char>(p);
13  return std::string(1, c);
14 }
15
```

Correct

Evaluation details:

Evaluation:

-Summary of tests

```
>+------------------------------+
>|  1  test run/ 1  test passed |
>+------------------------------+
```

```
enum struct Punctuation : char { Period='.', Colon=':', Semicolon=';',
                                 Question='?', Exclamation = '!', Comma = ',' };

std::string pun_to_str(Punctuation p) {
  switch(p) {
    case Punctuation::Period:      return std::string(".");
    case Punctuation::Colon:       return std::string(":");
    case Punctuation::Semicolon:   return std::string(";");
    case Punctuation::Question:    return std::string("?");
    case Punctuation::Exclamation: return std::string("!");
    case Punctuation::Comma:       return std::string(",");
    default:                       return std::string{};
  }
}
```

Question **5**

Correct

Mark 10.00 out of 10.00

Define a scoped enumeration type **Day** with enumerators [in this order] **Monday**, **Tuesday**, **Wednesday**, **Thursday**, **Friday**, **Saturday**, and **Sunday**.

Next, overload the prefix and postfix decrement operators and left-shift [or insert] operator that satisfies the following use cases:

```
Day today{Day::Monday};
std::cout << today;     // prints Monday to standard output stream
std::cout << --today;   // prints Sunday to standard output stream
std::cout << today--;   // prints Sunday to standard output stream
std::cout << today;     // prints Saturday to standard output stream
std::cout << --today;   // prints Friday to standard output stream
std::cout << ----today; // prints Wednesday to standard output stream
std::cout << --today;   // prints Tuesday to standard output stream
std::cout << --today;   // prints Monday to standard output stream
```

C++ standard library header **<ostream>** is included. Adding other standard library headers will prevent your code from compiling!!!

```
4  // Standard library header <ostream> is included. You cannot include any other header files!!!
5  // Define enumeration type Day with the appropriate enumerators and then
6  // overload prefix and postfix decrement operators and the left-shift [insert] operator ...
7  enum class Day
8  {
9      Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday
10 };
11
12 std::ostream& operator<<(std::ostream& os, const Day& day)
13 {
14     switch (day) {
15         case Day::Monday:
16             os << "Monday";
17             break;
```

Correct

Evaluation details:

Evaluation:

-Summary of tests

```
>+------------------------------+
>|  1  test run/ 1  test passed |
>+------------------------------+
```

```
enum class Day : int {Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday};

Day& operator--(Day& d) {
  d = (d == Day::Monday) ? Day::Sunday : Day{static_cast(d)-1};
  return d;
}

Day operator--(Day& d, int) {
  Day old{d};
  --d;
  return old;
}

std::ostream& operator<<(std::ostream& os, Day const& d) {
  switch (d) {
    case Day::Monday:    os << "Monday"; break;
    case Day::Tuesday:   os << "Tuesday"; break;
    case Day::Wednesday: os << "Wednesday"; break;
    case Day::Thursday:  os << "Thursday"; break;
    case Day::Friday:    os << "Friday"; break;
    case Day::Saturday:  os << "Saturday"; break;
    case Day::Sunday:    os << "Sunday"; break;
    default:             os << "Bad day!!!";
  }
  return os;
}
```

Question **6**

Correct

Mark 5.00 out of 5.00

Which of the following declares *a pointer to an array of function pointers*?

Select one:

- `void (*x[3])();`
- `void *x[3]();`
- `void (*x)[3]();`
- None of the listed choices
- `void (*(*x)[3])();` ✔

Your answer is correct.

The correct answer is: `void (*(*x)[3])();`

Question **7**

Correct

Mark 5.00 out of 5.00

Given the following definitions:

```
int f1() { return 16; }
int f2() { return 32; }
int f3() { return 64; }

int (*pf[])() {f1, f2, f3};
```

evaluate the expressions below.

`(*(pf+1))()`	32	✔
`(*&pf[2])()`	64	✔
`pf()`	Compile-time error	✔
`pf[1]()`	32	✔
`(*pf)()`	16	✔
`pf[2]`	Address of function f3	✔
`(*pf+2)()`	Compile-time error	✔
`(*(pf+1))`	Address of function f2	✔

The correct answer is: `(*(pf+1))()` → 32, `(*&pf[2])()` → 64, `pf()` → Compile-time error, `pf[1]()` → 32, `(*pf)()` → 16, `pf[2]` → Address of function f3, `(*pf+2)()` → Compile-time error, `(*(pf+1))` → Address of function f2

Question **8**

Correct

Mark 5.00 out of 5.00

Write the ***exact*** text written to the standard output stream by the following code fragment:

```
void print(int x, void (*printer)(int)) {
  (*printer)(x);
}

void printer1(int x) { std::cout << x + 1; }
void printer2(int x) { std::cout << 2*x; }
void printer3(int)   { std::cout << 1; }

// in function main ...
print(8, &printer3);
print(0, &printer1);
print(1, &printer2);
```

Answer: 112 ✔

The correct answer is: 112

Question **9**

Correct

Mark 10.00 out of 10.00

Define functions **f0**, **f1**, **f2**, **f3**, and **f4** that take no values and return **int** values 4, 8, 16, 32, and 48, respectively.

Next, define a function **foo** that returns a dynamically allocated array of 5 elements containing addresses of the previously defined functions **f0**, **f1**, **f2**, **f3**, and **f4**. The definition of function **foo** must satisfy the following use cases:

```
??? fp = foo(); // fp will point to the first element of the
                // dynamically allocated memory allocated by foo
std::cout << fp[0](); // prints 4 to standard output stream
std::cout << fp[1](); // prints 8 to standard output stream
std::cout << fp[2](); // prints 16 to standard output stream
std::cout << fp[3](); // prints 32 to standard output stream
std::cout << fp[4](); // prints 48 to standard output stream
delete[] fp; // return heap memory allocated by foo
```

No standard library headers are required and therefore none are included. Adding standard library headers will prevent your code from compiling!!!

```
// Standard library headers are not necessary and therefore none are included.
// Neither can you include any other header files!!!
// Define functions f0, f1, f2, f3, f4, and foo ...
using FP_INT = int (*)();

int f0() { return 4; }
int f1() { return 8; }
int f2() { return 16; }
int f3() { return 32; }
int f4() { return 48; }

FP_INT* foo() {
    FP_INT* p = new FP_INT[5];
    *(p + 0) = &f0;
```

Correct

Evaluation details:

Evaluation:

-Summary of tests

```
>+------------------------------+
>|  1  test run/ 1  test passed |
>+------------------------------+
```

```
int f0() { return 4; }
int f1() { return 8; }
int f2() { return 16; }
int f3() { return 32; }
int f4() { return 48; }

int (**foo())() {
  using FP_INT = int (*)();
  FP_INT *afp = new FP_INT [5] {f0, f1, f2, f3, f4};
  return afp;
}
```

Possible solution:

```
int f0() { return 4; }
int f1() { return 8; }
int f2() { return 16; }
int f3() { return 32; }
int f4() { return 48; }

int (**foo())() {
  using FP_INT = int (*)();
  FP_INT *afp = new FP_INT [5] {f0, f1, f2, f3, f4};
  return afp;
}
```

◄ Assignment 6: Doubly Linked List - Operator Overload

Jump to...

Lab: Doubly Linked List - Class Template ►