CPA Chapter 3 Practice Quiz
**Question:** What is the output of the following code fragment in C++? (assumption: all `#include` and the rest of the code are correct)

```cpp
int myInt1 = 3, myInt2 = 3;
int *pnt1 = &myInt1, *pnt2 = &myInt2;
myInt1 = ++(*pnt1) + (*pnt1);
myInt2++;
myInt2 = (*pnt2) + (*pnt2);
cout<< myInt1<<myInt2<<endl;
```

**Answers:**

A) 86  
B) 78  
C) 88  
D) 77
**Question:** What is the output of the following code fragment in C++

```cpp
void SetElements(int index, int **array, int value = 0) {
    (*array)[*(&index)] = value;
}

int main() {
    int *point1, *point2;
    point1 = new int[1];
    point2 = new int[2];
    *point1 = 0;
    SetElements(*&*point1, &point2);
    point1[0] = 1;
    SetElements(*&*point1, &point2, *point1);
    cout << point2[(*point1)] << point2[(*point1) - 1] << endl;
    delete[] point1;
    delete[] point2;
}
```

**Answers:**
- A) 11
- B) 10
- C) 01
- D) 00
<table>
<thead>
<tr>
<th>Subject: Functions overloading</th>
<th>Question Number: 3</th>
</tr>
</thead>
</table>

### Question

Which of the function call will generate a compiler error?

```cpp
void f(){}
void f(int){}
void f(float){}
void f(float, int = 0){}
int main(){
    short myShort;
    int myInt;
    unsigned int myUInt;
    float myFloat;
    double myDouble;
    f(myShort);
    f(myInt);
    f(myFloat);
    f(myUInt);
    f(myDouble);
}
```

### Answers

- **A**) f(myShort)
- **B**) f(myInt)
- **C**) f(myFloat)
- **D**) f(myUInt)
- **E**) f(myDouble)
### Question

The values of the following variables are $a = \ldots$, $b = \ldots$, $c=\ldots$.

They are... memory leaks in the program.

```cpp
void ModifyVariables(int a, int &b, int *c)
{
    a = b;
    b+=a;;
    c = new int(b);
    (*c)++;  
}

int main()
{
    int a=0, b=1, *c;
    c = new int(2);
    ModifyVariables(a,b,c);
    cout<<a<<b<<*c;
    delete c;
}
```

### Answers

[see question instructions above]
## Question
What does the following code fragment in C++ do? (assumption: all #include and the rest of the code are correct)

```cpp
void Pointer(int *p)
{ 
  (*p)++;  
  Reference(*p);  
  cout<<*p;  
}
void Reference(int &p)
{ 
  p++;  
  Value(p);  
  cout<<p;  
}
void Value(int p)
{ 
  p++;  
  cout<<p;  
}
int main()
{ 
  int value =3;  
  Pointer(&value); 
  cout<<value;  
}
```

### Answers:

- A) 6555
- B) 6565
- C) 5656
- D) 6565
### Question

What does the following code fragment in C++ display? (assumption: all #include and the rest of the code are correct)

```cpp
int main(){
    int *index;
    int * vector;
    int i =128;
    vector = new int[5];
    index =vector;
    while (i){
        if (i%2 == 0)
            *index = i;
        i/=2;
        if ((vector + 4) == index )
            break;
        index++;
    }
    cout<<vector[i%5]<<endl;
    delete[] vector;
}
```

### Answers:

A) 8  
B) 16  
C) 128  
D) 64
**ANSWER KEY**

<table>
<thead>
<tr>
<th>Correct answers:</th>
<th>Q1 - C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Explanation: C is correct because:</td>
<td></td>
</tr>
<tr>
<td>myInt1 = ++(*pnt1) + (*pnt1); // we first pre increment the value of pnt1 (since C++11 this behavior is not considered undefined) and then we add the values.</td>
<td></td>
</tr>
<tr>
<td>myInt2++; // sets the value of myInt2 to 4</td>
<td></td>
</tr>
<tr>
<td>myInt2 = (*pnt2)++ + (*pnt2); // pnt2 is a pointer to myInt2 (so when we dereference it returns 4 – the value of myInt2)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Correct answers:</th>
<th>Q2 - B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Explanation: B is correct because</td>
<td></td>
</tr>
<tr>
<td>SetElement function – sets the element from position “index” of the array “array” to value “value”: array[index] =value.</td>
<td></td>
</tr>
<tr>
<td>The first call sets the “point2[0]” to 0 and the second call sets “point2[1]” to 1</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Correct answers:</th>
<th>Q3 - C,D,E</th>
</tr>
</thead>
<tbody>
<tr>
<td>Explanation:C,D,E:</td>
<td></td>
</tr>
<tr>
<td>C – the call “f(myFloat)” is ambiguous to the compiler( two functions match: “f(float)” and “f(float, int =0)”</td>
<td></td>
</tr>
<tr>
<td>D,E – the compiler you cannot convert “uint” to “int” and “double” to “float”</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Correct answers:</th>
<th>Q4 - The values for the following variables are a = 0, b = 2, c = 2. They are 1 memory leaks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Explanation:</td>
<td></td>
</tr>
<tr>
<td>The variable “a” is transmitted by value (there is made a copy of the value of variable “a” so in the expression “a =b” is modified the value of the local variable “a” (from the function)</td>
<td></td>
</tr>
<tr>
<td>The variable “b” is transmitted by reference and therefore its value is modified</td>
<td></td>
</tr>
<tr>
<td>Even the variable “c” is a pointer, the pointer value itself is transmitted by value, so when you want “c” to point to another memory zone actually the copy made to the “c” pointer points to a new memory zone.</td>
<td></td>
</tr>
<tr>
<td>There is one memory leak because you are allocating memory in the “ModifyVariables” function that is never freed.</td>
<td></td>
</tr>
<tr>
<td>Correct answers:</td>
<td>Q5 - A</td>
</tr>
<tr>
<td>----------------</td>
<td>-------</td>
</tr>
<tr>
<td>Explanation: The “Value” function parameter is copied by value (so any changes made by the function to the variable are only local) The rest of the functions modify the content of the initial variable.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Correct answers:</th>
<th>Q6 - A</th>
</tr>
</thead>
<tbody>
<tr>
<td>Explanation: When we declare “new int[5]” the compiler allocates 5 consecutive blocks of memory of size int. “vector” points to the first element of the allocated memory block. “index++” goes to the next element from the block of memory (the next int) so “index+4” is the fifth element of the vector.</td>
<td></td>
</tr>
<tr>
<td>Chapter: 3</td>
<td>Extending the expressive power: pointers, functions and memory</td>
</tr>
<tr>
<td>------------</td>
<td>-------------------------------------------------------------</td>
</tr>
<tr>
<td>Section: 1</td>
<td>Pointers: another kind of data in the “C++” language</td>
</tr>
</tbody>
</table>

**C++ Associate (CPA)**

**Subject:** Pointers in C++

<table>
<thead>
<tr>
<th>Question type: single-choice</th>
</tr>
</thead>
</table>

**Question:** Which of the following statements is correct?

- A) Base class pointer cannot point to derived class.
- B) Derived class pointer cannot point to base class.
- C) Pointer to derived class cannot be created.
- D) Pointer to base class cannot be created.
### Question: Which of the following is correct about function overloading?

- A) The types of arguments are different.
- B) The order of arguments is different.
- C) The number of arguments is different.
- D) The return type of the function is same.
<table>
<thead>
<tr>
<th>Chapter: 3</th>
<th>Extending the expressive power: pointers, functions and memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Section: 1</td>
<td>Pointers: another kind of data in the “C++” language</td>
</tr>
<tr>
<td>C++ Associate (CPA)</td>
<td>Chapter: 3</td>
</tr>
<tr>
<td>Subject: Pointers in C++</td>
<td></td>
</tr>
</tbody>
</table>

**Question:** What is the implicit pointer that is passed as the first argument for non-static member functions?

<table>
<thead>
<tr>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>A) ‘self’ pointer</td>
</tr>
<tr>
<td>B) std::auto_ptr pointer</td>
</tr>
<tr>
<td>C) ‘Myself’ pointer</td>
</tr>
<tr>
<td>D) ‘this’ pointer</td>
</tr>
</tbody>
</table>
### Question

**Question:** Inline functions are invoked at the time of:

A) Run time  
B) Compile time  
C) Depends on how it is invoked
## ANSWER KEY

<table>
<thead>
<tr>
<th>Correct answers:</th>
<th>Q1 – B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Explanation:</td>
<td>no explanation</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Correct answers:</th>
<th>Q2 – A, B, C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Explanation:</td>
<td>Return type has nothing to do with method/function overloading</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Correct answers:</th>
<th>Q3 – D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Explanation:</td>
<td>no explanation</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Correct answers:</th>
<th>Q4 – B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Explanation:</td>
<td>no explanation</td>
</tr>
</tbody>
</table>
AUTHOR: VITALI KREMEZ  
VKREMEZ@HOTMAIL.COM

AUTHOR'S BIO: 
Becoming a programmer is deeply connected—the three year-long study of cybersecurity that students learn about on their first college day and do not stop thinking about until their last. It forces them to draw from all they have learned. It is my test of perseverance, creativity, and knowledge that appeared to be also, rather unexpectedly, the catalyst in my decision to study C++ programming. Vitali Kremez, CFE, CNDA, CEH, Sec+, Linux+, LPIC1, Suse CLA.

<table>
<thead>
<tr>
<th>Chapter: 3</th>
<th>Chapter 3: Extending the expressive power: pointers function and memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Section: 1.22</td>
<td>Another new operator</td>
</tr>
<tr>
<td>C++ Certified Programmer Associate (CPA)</td>
<td>Chapter: 3</td>
</tr>
<tr>
<td>Subject: 3.1.22 Another new operator</td>
<td></td>
</tr>
</tbody>
</table>

Question: Fill in the following blanks to declare an array of integers that has 19 elements. You should assign the value of 50 to each element using the for loop.

```cpp
#include <iostream>
using namespace std;

int main(){
    int cpp_int[100];
    for(int i =0; i<100; i++){
        cout << "The size of the 'array' is " << sizeof(cpp_int) << " bytes";
        return 0;
    }
}
```

Answers:
A. 100;  
B. 400;  
C. 800;  
D. 90
## Chapter 3: Extending the expressive power: pointers, functions and memory

### Section 3.2.2: Pointers vs. arrays

#### C++ Certified Programmer Associate (CPA)

<table>
<thead>
<tr>
<th>Chapter: 3</th>
<th>Chapter 3: Extending the expressive power: pointers, functions and memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Section: 2</td>
<td>Pointers vs. arrays</td>
</tr>
</tbody>
</table>

#### Question Type: Multiple-choice

**Subject:** Pointers vs. arrays

**Question Number:** 2

**Question:** What is the output of the code below?

```cpp
#include <iostream>
using namespace std;

int main() {
    int y[4] = { 77, 66, 55, 44 }; *ptr = y + 11;
    *(*ptr + 11))++;
    *ptr++;
    cout << y[1] << y[2];
    return 0;
}
```

**Answers:**

A. 7766  
B. 6655  
C. 5544  
D. 7755
## ANSWER KEY

<table>
<thead>
<tr>
<th>Correct answer:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1 - C. 400</td>
</tr>
<tr>
<td>Explanation:</td>
</tr>
<tr>
<td>N/A</td>
</tr>
<tr>
<td>Correct answer:</td>
</tr>
<tr>
<td>Q2 - B. 6655</td>
</tr>
<tr>
<td>Explanation:</td>
</tr>
<tr>
<td>N/A</td>
</tr>
</tbody>
</table>
AUTHOR: SRAJAN RATTI

AUTHOR'S BIO: I am a Final year student of Amity University India. I am doing my Engineering in BTECH CSE. I love to learn new languages. My hobbies are playing keyboard, making music, animation.

### Question

What would be the output of second cout?

```cpp
#include <iostream>

int main()
{
    int i = 5;
    int &r = i;
    std::cout << &r << "\n";
    r += 2;
    std::cout << &r << "\n";
    return 0;
}
```

#### Answers:

A. Output of both the cout would be same.
B. Value printed in second cout would be greater than value printed in first cout by 8.
C. Value printed in second cout would be lesser than value printed in first cout by 8.
D. Run Time or Compile Timer Error.
Question: What would be the output of the following code?

```cpp
#include <iostream>
int main()
{
    int i = 5;
    int *j = &i;
    void *x = j;
    std::cout << *x++;
    return 0;
}
```

Answers:

A. 6.
B. Error---> void * unknown size.
C. 5.
D. Garbage Value.
## Question
What does the following function declaration mean?

```c
char *(*abc)( float *, float **);
```

## Answers

A. abc is a pointer to a char taking pointer to a float and pointer to a pointer to a float.
B. Error.
C. abc is a function taking pointer to a float and pointer to a pointer to a float and returning pointer to a pointer to a char.
D. abc is a pointer to a function taking pointer to a float and pointer to a pointer to a float as argument and returning a pointer to char.
## Question:
What is the output of the following code?

```cpp
#include <iostream>

class XX{
   int is;
   public:
   XX(int x){
      is = x;
   }
   void display(){
      std::cout << "XX";
   }
};

class YY{
   int is;
   public:
   YY(int x){
      is = x;
   }
   void display(){
      std::cout << "YY";
   }
};

XX f(XX a){
a.display();
return a;
}YY f(YY b){
b.display();
return b;}

int main()
{f(5);
}
```

### Answers:

A. XX
B. YY
C. XXXY
D. YYXX
E. Error
**Chapter:** [3]  
**Extending the expressive power: pointers, functions and memory**

**Section:** [5]  
**Transferring data to and from functions**

|----------------------------------------|--------------|--------------|----------------------------------|

**Subject:** [Pointers to Functions]  
**Question Number:** [5]

**Question:** What is the output of the code?

```cpp
#include <iostream>

int yO(int xx)
{
    std::cout << xx;
    return xx;
}

int main()
{
    int(*__foos$)(int);
    __foos$ = &yO;
    (*__foos$)(100);
}
```

**Answers:**

A. Error at line 3.  
B. Error at line 4 and 5.  
C. 100.  
D. Address of variable.
<table>
<thead>
<tr>
<th>Chapter: [3]</th>
<th>Extending the expressive power: pointers, functions and memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Section: [4]</td>
<td>Declaring and defining functions</td>
</tr>
</tbody>
</table>

**Subject:** [Function syntax]

**Question Number:** [6]

**Question:** What is the syntax of `strncat` function?

// _Dest is Destination String and _Source is SourceString and num is number of //character to be copied

A. `strncat(char *Dest, const char *Source, size_t num);`
B. `strncat(size_t num, char *Dest, const char *Source);`
C. `strncat(const char *Source, char *Dest, size_t num);`
D. `strncat(size_t num, const char *Source, char *Dest);`

**Answers:**

A. A is correct
B. B is correct.
C. C is correct.
D. D is correct.
Question: What is the output of the following code?

```
#include <iostream>

int main() {
    char s = 'a';
    decltype((s)) bx = s;
    bx++;
    std::cout << "\n" << bx;
}
```

Answers:

A. a.
B. b.
C. c.
D. Error
<table>
<thead>
<tr>
<th>Correct answers:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1 - A.</td>
<td></td>
</tr>
</tbody>
</table>

**Explanation:** Value of reference Cannot be changed.

<table>
<thead>
<tr>
<th>Correct answers:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Q2 - A.</td>
<td></td>
</tr>
</tbody>
</table>

**Explanation:** The size of the object pointed to is unknown.

<table>
<thead>
<tr>
<th>Correct answers:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Q3 - D.</td>
<td></td>
</tr>
</tbody>
</table>

**Explanation:** Use spiral Rule to answer the question.  

<table>
<thead>
<tr>
<th>Correct answers:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Q4 - E.</td>
<td></td>
</tr>
</tbody>
</table>

**Explanation:** Clearly there is an ambiguity while calling function f(5)

<table>
<thead>
<tr>
<th>Correct answers:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Q5 - C.</td>
<td></td>
</tr>
</tbody>
</table>

**Explanation:**

<table>
<thead>
<tr>
<th>Correct answers:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Q6 - A.</td>
<td></td>
</tr>
</tbody>
</table>

**Explanation:**

<table>
<thead>
<tr>
<th>Correct Answer:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Q7 – C</td>
<td></td>
</tr>
</tbody>
</table>

**Explanation:**