A506 / C201 Computer Programming II
Placement Exam Sample Questions

For each of the following, choose the most appropriate answer (2pts each).

_____ **1**. Which of the following functions is causing a temporary object to be created?

```
class Date {...};
```

a. `Date Copy (Date &data);`
b. `void Add(Date &to, Date from);`
c. `void Add(Date &to, const Date from);`
d. All of the above

_____ **2**. A function is overloaded in the derived class when
a. a function from the base class is called on an object of the derived class;
b. the derived class implements a function with the same name and parameters as the base class
c. the derived class implements a function with the same name
d. none of the above

_____ **3**. Which of the following applies to a class derived from more than one class?
a. multiple inheritance
b. the compiler doesn't accept that
c. simple inheritance
d. polymorphism

_____ **4**. When do we need to have reference parameters for a function?
a. When the function needs to change the value of that parameter.
b. When the function needs to return more than one value.
c. When we don't want the function to make a local copy of an object.
d. All of the above.

_____ **5**. Which of the following operators should *not* be declared as friend?
a. >>
b. <<
c. +=
d. +

6. Write a function that will accept an integer array, we can assume global constant of MAX = 5, and then pass back the average of the numbers and the first number in the array. The function cannot use reference parameters to pass any thing back, so you need to use pointers. Do not need to comment the function or the code. Also write the bit of the main function that shows how the function is called.

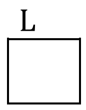
main here ***function here***

```
#include <iostream>
using namespace std;
const int MAX = 5;
void main()
{
   float Average; float First;
   float N[MAX] = {5,6,8,3,6};


}
```

7. Draw the linked structure that results from executing the code segment below. Do not include Temp in your final drawing.

```
struct Node
{
   char ch;
   Node *link;
};
```



```
Node *L, *temp;
L = new Node;
L->ch = '2';
L->link = new Node;
L->link->ch = '3';
temp = new Node;
temp->ch = '1';
L->link->link = temp;
temp->link = L;
```

8. In the same structure, implement the following function:

```
// Finds the last node and returns a pointer to it.
// If the list is empty, it returns NULL.
Node *Last(Node *front)
{
```

9. Write a program that will accept command line arguments. If there are no arguments, other than the invoking file name, then return 0. If there is one argument after the file name, return the square of this integer. We will assume that there will be no more than this number of arguments and also that the second argument will be a character in the range '0' to '9'.

10. What happens when you dereference a NULL pointer?

11. Assume that the pointer "a" was allocated the following way. Write an instruction that deallocates it.

```
float *a;
a = new float[10];
```

12. In the following recursive function, identify the base case and the recursive call.

```
long factorial(int n)
{
   if (n < 2)
    return 1;
   else
      return n * factorial(n-1);
}
```

13. Complete the following program, so that the program could display the required results.

Note: You need to allocate space if needed.

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

class Book
{
private:
      string title;         //every book has a title
      int editions;
      double * price;  //every book has a price


public:
      Book(string, double); // a single edition, one price
      Book(string, double*, int count); // multiple editions
      ~Book();
      string getTitle();
      double getPrice(int which=0);
      double getLowestPrice();
      virtual double get_total_cost()=0;
};
```

```
// Constructor of Book, assign parameters aTitle and aPrice to Book
// attributes title and price, respectively. Single edition.
Book::Book(string aTitle, double aPrice)
{
      // Your code goes here

}


// Constructor of Book with multiple editions with several prices.
Book::Book(string aTitle, double *prices, int count)
{
      // Your code goes here

}


//destructor of Book (1 point)
Book::~Book()
{
      // Your code goes here
}


string Book::getTitle()
{
      return title;
}

double Book::getPrice(int which)
{
      if (0 <= which && which < editions)
            return price[which];
      else
            exit(1); // quit the program with an error message
}

double Book::getLowestPrice()
{
      // Your code goes here


}
```

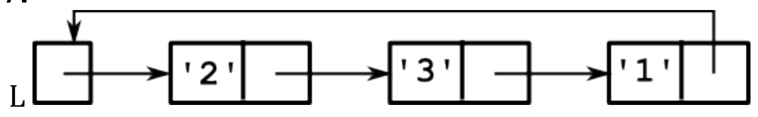
Answers

1. d
2. b
3. a
4. d
5. c

6.

main here	*function here*
`#include <iostream>`	`void ComputeAverage(int a[],`
`using namespace std;`	`    int *averagePtr,`
`const int MAX = 5;`	`    int firstPtr)`
`void main()`	`{`
`{  float average; float first;`	`   float sum = 0;`
`   float N[MAX] = {5,6,8,3,6};`	`   for (int i=0; i<MAX; i++)`
	`      sum += a[i];`
`   ComputeAverage(N, &average,`	`   *averagePtr = sum/MAX;`
`                  &first);`	`   *firstPtr = a[0];`
`}`	`}`

7.



8.
```
Node *Last(Node *front)
{
   if (front == NULL)
       return NULL;
   Node *lookup = front;
   while (lookup->link) {
      front = front->link;
   return front;
}
```

9.
```
int main(int argc, char **argv)
{
   if (argc == 1)
      return 0;
   else
   {
      int n = argv[1][0]-'0';
      return n*n;
   }
}
```

10. You get a runtime error or segmentation fault or memory access violation, depending on the operating system.

11. `delete[] a;`

12.
```
long factorial(int n)
{
   if (n < 2)  // ← base case
    return 1;
   else
      return n * factorial(n-1);
}                // ^recursive call
```

13.
```
Book::Book(string aTitle, double aPrice)
{
      title = aTitle;
      editions = 1;
      price = new double;
      *price = aPrice;
}

Book::Book(string aTitle, double *prices, int count)
{
      title = aTitle;
      if (count > 0)
      {
            editions = count;
            price = new double[count];
            for (int i=0; i<count; i++)
                  price[i] = prices[i];
      }
}

Book::~Book()
{
      if (editions > 0)
            delete[] price;
}

double Book::getLowestPrice()
{
      if (editions == 0)
            return 0;
      double minPrice = price[0];
      for (int i=1; i<editions; i++)
            if (price[i] < minPrice)
                  minPrice = price[i];
      return minPrice;
}
```