Practice Exam: Time 3 Hours [See your watch]

Question: Solve these questions and think thoroughly before seeking for help. Note that we can write a float constant as (2.0 or 2.)

Question 1:

1.1

What is the output of the following statements?

```
cout << 5. / 10 << endl;          __________
cout << 5 / 10 << endl;           __________
cout << 1 / 2 * 10 << endl;       __________
cout << 1. / 2 * 10 << endl;      __________
cout << 1 / 2. * 10 << endl;      __________
cout << 5 % 10 << endl;           __________
cout << -5 % 10 << endl;          __________
```

1.2

Which of the following lines are syntactically valid declarations of variables, constants, or arrays?

```
double xout = .99;
const int hundred = "100";
char alarm = \a;
const double int secs_in_day = 86400;
apvector<int> price(3) = 69, 79, 99;
double c-out = 2;
unsigned double freq = 166.;
```

1.3

Simplify:

```
if (!(x == 7) && !(x > 7))
    ...
```

```
bool inside = !((x < left) || (x > right) || (y < top)
    || y > bottom));
```

1.4

Which of the following lines are syntactically valid declarations?

```
double amt, *pAmt = 0;
char *str, ch = &str;
int n, r = &n;
char ch, &rch = ch, *pch = &ch;
double z, &z1 = z, &z2 = z;
```

Question 2: What will be the outputs of following snippets, identify any errors.
2.1

```
const short days = 365, hours = 24, mins = 60, secs = 60;
cout << "Seconds in a year = "
<< hours * mins * secs * double(days) << endl;
```

2.2

```
const double g = 16.;
double t;
cout << " Enter time in secs: ";
cin >> t;
cout << "The travel distance is " << 1 / 2 * (g * t * t)
<< endl;
```

2.3

```
short x, y;

cout << "Enter two integers x and y: ";
cin >> x >> y;
x *= double(x);
y *= double(y);
double sq_radius = x + y;
if (sq_radius > 250000.)
    cout << "(x,y) is outside the circle.\n";
```

2.4

```
int c = 3;
cout << c++;
cout << c;
cout << ++c;
cout << c << endl;
```

_____

2.5

```
int c = 3;
c += 4; c /= 4; c -= 1; c *= 4;
cout << "c = " << c << endl;
```

2.6

```
#include <iostream.h>

int main()

{
    int count, max = 6;
    for (count = 1;   count < max;   count++)
        cout << ++count;
    cout << endl;
    return 0;
}
```

2.7

```
int main()

{
    int count = 0;

    while (count <= 1) count += 0.1;
    cout << count << endl;
    return 0;
}
```

2.8

```
string hello=" hello";
```

```
int i = 1;
while (i < 6) {
hello[i-1] = hello[i];
i++;
}
hello[5] = '!';

cout<< hello << endl;
```

Output:

2.9

```
const int SIZE = 8;

int a[SIZE];
for (int i = 0; i < SIZE/2; i++)
     a[2*i] = i + 1;
for (int i = SIZE/2; i < SIZE; i++)
     cout << a[i] << ' ';
cout << endl;
```

2.10

```
char ch1 = '*', ch2 = '+', *s = &ch1;
ch2 = *s;
cout << ch1 << ch2;
```

2.11

```
int x = 3, y = 9, &r = y, *p;
p = &y;
*p = 0;
cout << x << r;
```

2.12

```
double u = 1.1, v = 1.2, *max = &u;
if (v > u) max = &v;
cout << *max;
```

2.13

```
char *yes = "yes";
if (*yes == 'y') yes = "no";
cout << yes << endl;
```

2.14

```
double z = 2000, &century = z;
cout << century++;
```

2.15

```
int a = –7, b = 3, *p = &a;
cout << *p;
p = &b;
cout << *p << endl;
```

2.16

```
int i = 5, j = 6, k = 7, n = 3;
cout << i + j * k - k % n << endl;
cout << i / n << endl;
```

2.17

```
int found = 0, count = 5;
if (!found || --count == 0)
cout << "danger" << endl;
cout << "count = " << count << endl;
```

2.18

```
char ch;
char title[] = "Titanic";
ch = title[1];
title[3] = ch;
cout << title << endl;
cout << ch << endl;
```

2.19

```
int main()
{
 cout  <<  (n = 4)  <<  endl;
 cout  <<  (n == 4)  <<  endl;
 cout  <<  (n > 3)  <<  endl;
 cout  <<  (n < 4)  <<  endl;
 cout  <<  (n = 0)  <<  endl;
 cout  <<  (n == 0)  <<  endl;
 cout  <<  (n > 0)  <<  endl;
 cout  <<  (n && 4)  <<  endl;
 cout  <<  (n || 4)  <<  endl;
 cout  <<  (!n)  <<  endl;

 }
```

2.20

```
main()
{
int n = 3;
while (n >= 0)
{
cout << n * n << endl;
--n;
}
cout << n << endl;
while (n < 4)
cout << ++n << endl;
cout << n << endl;
while (n >= 0)
cout << (n /= 2) << endl;
```

```
     return 0;
     }
```

2.21

```
main()
{
int n = 4, k = 2;
cout << ++n << endl;
cout << n << endl;
cout << n++ << endl;
cout << n << endl;
cout << -n
 << endl;
cout << n
 << endl;
cout << --n << endl;
cout << n << endl;
cout << n-- << endl;
cout << n << endl;
cout << n + k << endl;
cout <<
 n
 << endl;
cout <<
 k
 << endl;
cout << n << k << endl;
cout <<
 n << endl;
cout << "
 " << n << endl;
cout << " n" << endl;
cout << "\n" << endl;
cout << " n * n = "; //CAREFUL!
cout << n * n << endl;
cout << 'n' << endl;
return 0;
}
```

Question 3:

3.1

Write a function:

```
double BodyMassIndex(int inches, int lbs);
```

that takes a person's height in inches and weight in pounds as arguments and returns the body mass index (BMI). BMI is defined as the weight, expressed in kilograms, divided by the square of the height expressed in meters. One inch is 0.0254 meters and one pound is 0.454 kilograms. Write a program that prompts the user for his weight and height, calls `BodyMassIndex(…)`, and displays the BMI.

3.2
Write a function Distance that returns the distance between the points (x1,y1) and (x2,y2). The distance formula is
d = sqrt((x2 - x1)2 + (y 2 - y 1)2);
and the function should work for all integer values of coordinates.

Fill in the blanks in the following functions:

```
bool isdigit(char d)

// Returns true if d is a digit (in ASCII code), false otherwise.

{
    return  _____________________________________________
}


bool isalpha(char c)

// Returns true if c is a letter (in ASCII code),
//   false otherwise.

{
    return   ____________________________________________

             ____________________________________________
}
```

3.3

Write a function named "***subtotal***" takes as its arguments the following:

(1) an array of floating point values;

(2) an integer that tells the number of cells in the array.

The function should replace the contents of each cell with the sum of the contents of all the cells in the original array from the left end to the cell in question. Thus, for example, if the array passed to the function looks like this:

0	1	2	3	4
5.8	2.6	9.1	3.4	7.0

then when the function returns, the array will have been changed so that it looks like this:

0	1	2	3	4
5.8	8.4	17.5	20.9	27.9

because 5.8 + 2.6 = 8.4 and 5.8 + 2.6 + 9.1 = 17.5 and so on. Note that the contents of cell 0 are not changed. The function should not return a value.

3.4

The sum

$$1-\frac{1}{2}+\frac{1}{3}-\frac{1}{4}+\ldots-\frac{1}{n}+\ldots$$

converges to ln 2 — the natural log of 2 . Write a program that prompts the user for a positive integer *n*, estimates ln 2 by adding the first *n* terms of the sequence, displays the estimate together with the value of `log(2.)` returned by the library function `double log(double x)`, declared in `math.h`. Note that in C++, `log(...)` returns the natural log and `log10(...)` returns the logarithm base 10.

3.5

```
Write following function
bool HasPunctuation(const string &str)
// Returns true if str includes a period, a comma,
//  a semicolon, or a colon; false otherwise.
```

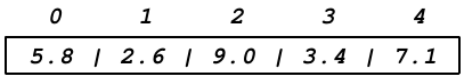
3.6

Write a function named "***reverse***" that takes as its arguments the following:

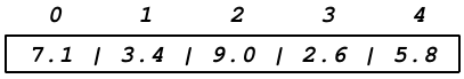
(1) an array of floating point values;

(2) an integer that tells how many floating point values are in the array.

The function must reverse the order of the values in the array. Thus, for example, if the array that's passed to the function looks like this:



then when the function returns, the array will have been modified so that it looks like this:



The function should not return any value.

3.7

Write a function named "***sum***" that takes as its arguments the following:

(1) an array of floating point values;

(2) an integer that tells how many floating point values are in the array.

The function should return as its value the sum of the floating point values in the array. Thus, for example, if the array that's passed to the function looks like this:

0	1	2	3	4
5.8	2.6	9.0	3.4	7.1

then the function should return the value 27.9 as its value.

3.8

Write a function named "***eliminate_duplicates***" that takes an array of integers in random order and eliminates all the duplicate integers in the array. The function should take two arguments:

(1) an array of integers;

(2) an integer that tells the number of cells in the array.

The function should not return a value, but if any duplicate integers are eliminated, then the function should change the value of the argument that was passed to it so that the new value tells the number of distinct integers in the array. Here is an example. Suppose the array passed to the function is as shown below, and the integer passed as an argument to the function is 11.

0	1	2	3	4	5	6	7	8	9	10
58	26	91	26	70	70	91	58	58	58	66

Then the function should alter the array so that it looks like this:

0	1	2	3	4	5	6	7	8	9	10
58	26	91	70	66	??	??	??	??	??	??

and it should change the value of the argument so that it is 5 instead of 11 . The question marks in the cells after the 5th cell indicate that it does not matter what numbers are in those cells when the function returns.