

1. control structures

Below are some practice exam questions on **control structures**. Answer the questions on paper first, and then enter them into the computer and run them to check your answers.

(a) What is the exact output of the following program:

```
#include <iostream>
using namespace std;
int main() {
for ( int i=1; i<5; i++ ) {
    cout << "i = [" << i << "]\n";
}
    } // end of main()
```

(b) What is the exact output of the following program:

```
#include <iostream>
using namespace std;
int main() {
    double max = 9.876;
    int counter = 0;
    while ( counter < max ) {
    cout << counter << " ";
    counter += 3;
    }
    cout << "\nthat's all folks!\n";
} // end of main()
```

(c) What is the exact output of the following program:

```
#include <iostream>
using namespace std;
int main() {
    double pie = 3.14;
    int pizza = 8;
    if ( pizza / 2 > pie ) {
        cout << "there's enough for 2 people\n";
    }
    else {
        cout << "we'll have to order more\n";
    }
} // end of main()
```

2. strings

Below are practice exam questions on **strings**. Answer the questions on paper first, and then enter them into the computer and run them to check your answers.

(a) What is the EXACT output of the program below?

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

void doSomething( string s ) {
    int i = 0;
    while ( ( i < s.length() ) && ( s.at( i ) != ' ' ) ) {
        cout << (char)toupper( s.at( i ) );
        i++;
    }
    cout << s.substr( i, s.length()-i ) << endl;
} // end of doSomething()

int main() {
    string greeting = "hello world";
    doSomething( greeting );
} // end of main()
```

- (b) In the above program, if the variable `greeting` defined in `main()` were set to "now is the time" instead of "hello world", what would the output of the program be?
- (c) In the above program, what does the function `doSomething()` do, in general?

3. functions

Below is a practice exam question on **functions**. Answer the question on paper first, and then enter your answer into the computer and run it to check your answers.

Imagine that you have a sprite that wanders around in a 2-dimensional world made up of ASCII characters. Some of the characters in its world are letters, some are numbers, some are punctuation marks, some are non-printable characters. The sprite knows which ASCII character is at its (x, y) location. If the sprite is at a location where there is a letter ('A'...'Z' or 'a'...'z'), then that's cool and the sprite is happy. If the sprite is at a location where there is a digit ('0'...'9'), then that is a special treasure, and the sprite is very happy. But if the sprite is standing on any non-alphanumeric character, then that is considered dangerous and the sprite is not happy. Below is the skeleton of a program that simulates this sprite.

```
#include .....
using namespace std;

// function prototypes
void initWorld();
bool isDanger( char c );
bool isTreasure( char c );
int valueOfTreasure( char c );

// global variables
const int MAX = 100;
char world[100][100];

// main function
int main() {
    .....
} // end of main()

// initialize the sprite's world
void initWorld() {
    for ( int i=0; i<MAX; i++ ) {
        for ( int j=0; j<MAX; j++ ) {
            world[i][j] = (char)( rand() % 128 );
        }
    }
} // end of initWorld()
```

Complete the program as follows:

- Fill in the missing "include" files at the top of the program.
- Write the `isDanger()` function. It should return `true` if the argument `c` is NOT a letter or a digit; or `false` otherwise.
- Write the `isTreasure()` function. It should return `true` if the argument `c` is a digit ('0'...'9'); or `false` otherwise.
- Write the `valueOfTreasure()` function. It should return the numeric value of the argument `c`. Remember that "treasure" is represented by the digits '0' through '9'. These are ASCII digits, so their numeric (integer) values are not the same as their character values. For example, the character '0' has

an ASCII code value of 48 and the character '5' has an ASCII code value of 53. To convert an ASCII digit to its equivalent numeric value, you subtract 48 from it. The code below shows how to convert the two examples:

```
char myChar = '0';
char anotherChar = '5';
int myNumber;
int anotherNumber;
myNumber = (int)myChar - 48;
anotherNumber = (int)anotherChar - 48;
```

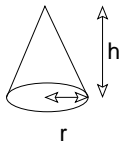
- Write the `main()` function:

- Initialize the random number seed.
- Call the `initWorld()` function to initialize the sprite's world.
- Compute the following statistics about the sprite's world:
 - * the number of locations that are dangerous
 - * the number of locations that have treasure
 - * the number of locations that are cool (i.e., not dangerous and no treasure)
 - * the numeric total of all the treasure locations

Hint: create two nested loops that will cycle through all the locations in the sprite's world and call the functions you wrote above to assess each location.

4. using **value** parameters

- (a) Write a function that takes two call-by-value double parameters and computes and returns the volume (v) of a cone using the following equation:



The diagram shows a cone with a circular base. A horizontal double-headed arrow across the base is labeled 'r', representing the radius. A vertical double-headed arrow from the center of the base to the apex is labeled 'h', representing the height.

$$v = (1/3)\pi r^2 h$$

The function prototype looks like this:

```
double coneVolume( double r, double h ) {  
    ...  
} // end of coneVolume()
```

- (b) Write a `main()` function that asks the user to enter the radius of the base of the cone (r) and the height of the cone (h). Echo the user's input. Then call your function to compute the volume and output the result.

HINTS:

- Remember that if you compute $1/3$ in C++, it will consider 1 and 3 to be integers, and it will perform *integer division*, which means that $1/3 = 0$ instead of 0.33333. In order to force C++ to do math with real numbers (i.e., doubles), use $1.0/3.0$ instead of $1/3$.
- Don't forget that π is a *constant* in the `cmath` library: `M_PI`.

5. using **reference** parameters

- (a) Modify the program you wrote (above) to include a “swap” function that exchanges the values of its two double arguments. The function prototype looks like this:

```
void swap( double &a, double &b ) {  
    ...  
} // end of swap()
```

- (b) Modify your `main()` function so that after doing what you did above, you then call the `swap()` function to swap the values of the radius of the base and the height of the cone that the user entered (i.e., the radius becomes the height and the height becomes the radius).
- (c) Then display the new values for radius and height, and call the `coneVolume()` function again with the new values. Display the result.

6. global variables

Write a program that does the following:

- Declare an array of 12 integers as a *global variable*. (Recall that this means that the array is declared outside of the `main()` function, and not inside any other function.)
- Write a function that initializes the value of each array element to the same value as its index. This means that, if my array is named `A`, then the value of `A[0]` will be 0, and the value of `A[1]` will be 1, etc. The function prototype looks like this:

```
void initArray();
```

- Write a function that displays the value of each array element. The function prototype looks like this:

```
void displayArray();
```

- Write a function that adds a random number to each entry in the array. The function prototype looks like this:

```
void muddleArray();
```

- Write a function that returns the value of the *largest* entry in the array. The function prototype looks like this:

```
int largestInArray();
```

- Write a function that returns the *average* value stored in the array. The function prototype looks like this:

```
double arrayAverage();
```

- Write the `main()` function, which should do the following:
 - (1) initialize the array;
 - (2) display the array;
 - (3) find and display the value of the largest element in the array;
 - (4) find and display the average of the values in the array;
 - (5) muddle the array;
 - (6) display the array (again);
 - (7) find and display the value of the largest element in the array (again);
 - (8) find and display the average of the values in the array (again)