Programming Exercises for coding in C and C++.

**Exercise 1 :**

Write a C++ program to calculate Fubanaci numbers Xn, n=1,2, …,1000

Xn+1 = Xn+ Xn-1, X0=1, , X1=1, n=1,2,…

**Exercise 2:**

Write a C++ program TO REARRANGE A LIST OF RANDOUM NUMBERS Xn INTO ASCENDING ORDER

**Exercise 3:**

Write a C++ program to get mean of the degrees of student

( Math = 80 , Science =67, Arabic= 65 , Studies= 88 , English = 56.

**Exercise 4:**

Write a C++ program to transfer inchs to centimeters (1 inch= 2.54 cintimeters)

**Exercise 5:**

-Use for loop to write a program to compute the factorial of a number

-Write another version to calculate the factorial recursively

**Exercise 6** :

Write a C++ program to get the volume and the area of a sphere or radius r

**Exercise 7**:

Write a C++ program to get the volume and the area of a cylinder of radius r and height h

**Exercise 8**:

Write a C++ program to get the sum of the series

SUM : 1-1/2 +1/3 -1/4 +…..+1 /99- 1 /100 + ...1/N, where N is user defined

**Exercise 9** :

Write a C++ program to get the sum of the series

SUM : 1-1/2! +1/3 ! - 1/4! +…..+1 /99!- 1 /100! + ...1/N!, where N is user defined

**Exercise 10**:

Write a C++ program to get the sum of the series

**Exercise 11:**

Write a C++ program to solve the second degree equation

A X2+B X + C = 0, where the inputs are the coefficients A, B and C 2

**Exercise 12**:

Write a C++ program to inter N random values between 0 and 100 and get their mean and the standard deviation *sd*

(*research for formulas of mean and standard deviation)*

Exercise 14 :

Write a C++ program to inter the elements of the two matrix A and B and their sum and difference

**Exercise 15-** Write a C++ program to solve the difference equation

Exercise 16- Write a C++ program to get the result of the multiplication of two matrices A and B

Exercise 17-

Write a C++ program to calculate the sum of the series

**Exercise 18:**

Write a program to convert English units to metric (e.g., miles to kilometers, gallons to liters, etc.). Include a specification and a code design.

**Exercise 19:**

Write a program to perform date arithmetic, such as how many days there are between 6/1/90 and 8/3/92. Include a specification and a code design. 3

**Exercise 20:**

A serial transmission line can transmit 960 characters a second. Write a program that will calculate how long it will take to send a file, given the file's size. Try it on a 400MB (419,430,400 byte) file. Use appropriate units. (A 400MB file takes days.)

**Exercise 21:**

--Write a program to tell whether a number is prime.

--Write another program to list all prime numbers between 1 and n, where n given by the user

**Exercise 22:**

Write a program that takes a series of numbers and counts the number of positive and negative values.

**Exercise 23:**

**Exercise 24:**

Write a c++ program to get the sum of a square of odd numbers **between** A and B where A and B are given by the user

**Exercise 25:**

Write a C++ program to enter names and degrees of student and give the grades according to the degree

**Exercise 26:**

Write a C++ program to calculate the area of a triangle with sides a,b and c

**\*** 4

PART 2:

**23.** Write a function named "g\_c\_d" that takes two *positive* integer arguments and returns as its value the

greatest common divisor of those two integers. If the function is passed an argument that is not positive (i.e.,

greater than zero), then the function should return the value 0 as a sentinel value to indicate that an error

occurred. Thus, for example,

cout << g\_c\_d(40,50) << endl; // will print 10

cout << g\_c\_d(256,625) << endl; // will print 1

cout << g\_c\_d(42,6) << endl; // will print 6

12

cout << g\_c\_d(0,32) << endl; // will print 0 (even though 32 is the g.c.d.)

cout << g\_c\_d(10,-6) << endl; // will print 0 (even though 2 is the g.c.d.)

13

**24.** A positive integer n is said to be ***prime*** (or, "a prime") if and only if n is *greater than* 1 and is divisible

only by 1 and n . For example, the integers 17 and 29 are prime, but 1 and 38 are not prime. Write a

function named "is\_prime" that takes a *positive* integer argument and returns as its value the integer 1 if

the argument is prime and returns the integer 0 otherwise. Thus, for example,

cout << is\_prime(19) << endl; // will print 1

cout << is\_prime(1) << endl; // will print 0

cout << is\_prime(51) << endl; // will print 0

cout << is\_prime(-13) << endl; // will print 0

**25.** Write a function named "digit\_name" that takes an integer argument in the range from 1 to 9 ,

inclusive, and prints the English name for that integer on the computer screen. No newline character should be

sent to the screen following the digit name. The function should not return a value. The cursor should remain on

the same line as the name that has been printed. If the argument is not in the required range, then the function

should print "digit error" without the quotation marks but followed by the newline character. Thus, for

example,

the statement digit\_name(7); should print seven on the screen;

the statement digit\_name(0); should print digit error on the screen and place

the cursor at the beginning of the next line.

**26.** Write a function named "reduce" that takes two positive integer arguments, call them "num" and

"denom", treats them as the numerator and denominator of a fraction, and reduces the fraction. That is to say,

each of the two arguments will be modified by dividing it by the greatest common divisor of the two integers.

The function should return the value 0 (to indicate failure to reduce) if either of the two arguments is zero or

negative, and should return the value 1 otherwise. Thus, for example, if m and n have been declared to be

integer variables in a program, then

m = 25;

n = 15;

if (reduce(m,n))

cout << m << '/' << n << endl;

else

cout << "fraction error" << endl;

will produce the following output:

5/3

Note that the values of m and n were modified by the function call. Similarly,

m = 63;

n = 210;

if (reduce(m,n))

cout << m << '/' << n << endl;

else

cout << "fraction error" << endl;

14

will produce the following output:

3/10

Here is another example.

m = 25;

n = 0;

if (reduce(m,n))

cout << m << '/' << n << endl;

else

cout << "fraction error" << endl;

will produce the following output:

fraction error

The function reduce is allowed to make calls to other functions that you have written.

**27.** Write a function named "swap\_floats" that takes two floating point arguments and interchanges the

values that are stored in those arguments. The function should return no value. To take an example, if the

following code fragment is executed

float x = 5.8, y = 0.9;

swap\_floats (x, y);

cout << x << " " << y << endl;

then the output will be

0.9 5.8

**28.** Write a function named "sort3" that takes three floating point arguments, call them "x" , "y" , and "z"

, and modifies their values, if necessary, in such a way as to make true the following inequalities: x \_ y \_ z .

The function should return no value. To take an example, if the following code fragment is executed

float a = 3.2, b = 5.8, c = 0.9;

sort3 (a, b, c);

cout << a << " " << b << " " << c << endl;

then the output will be

0.9 3.2 5.8

The function sort3 is allowed to make calls to other functions that you have written.

15

**29.** 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:

0 1 2 3 4

5.8 | 2.6 | 9.0 | 3.4 | 7.1

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

0 1 2 3 4

7.1 | 3.4 | 9.0 | 2.6 | 5.8

The function should not return any value.

**30.** 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.

**31.** Write a function named "location\_of\_largest" that takes as its arguments the following:

(1) an array of integer values;

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

The function should return as its value the subscript of the cell containing the largest of the values in the array.

Thus, for example, if the array that's passed to the function looks like this:

0 1 2 3 4

58 | 26 | 90 | 34 | 71

then the function should return the integer 2 as its value. If there is more than one cell containing the largest of

the values in the array, then the function should return the *smallest* of the subscripts of the cells containing the

largest values. For example, if the array that's passed to the function is

0 1 2 3 4 5 6

58 | 26 | 91 | 34 | 70 | 91 | 88

then the largest value occurs in cells 2 and 5 , so the function should return the integer value 2 .

16

**32.** Write a function named "location\_of\_target" that takes as its arguments the following:

(1) an array of integer values;

(2) an integer that tells how many integer values are in the array;

(3) an integer "target value".

The function should determine whether the given target value occurs in any of the cells of the array, and if it

does, the function should return the subscript of the cell containing the target value. If more than one of the cells

contains the target value, then the function should return the largest subscript of the cells that contain the target

value. If the target value does not occur in any of the cells, then the function should return the sentinel value 1

. Thus, for example, if the target value that's passed to the function is 34 and the array that's passed to the

function looks like this:

0 1 2 3 4 5 6

58 | 26 | 91 | 34 | 70 | 34 | 88

then the target value occurs in cells 3 and 5 , so the function should return the integer value 5 .

**33.** Write a function named "rotate\_right" that 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 shift the contents of each cell one place to the right, except for the contents of the last cell,

which should be moved into the cell with subscript 0 . 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

7.0 | 5.8 | 2.6 | 9.1 | 3.4

The function should not return a value.

**34.** Write a function named "shift\_right" that takes as its arguments the following:

(1) an array of floating point values;

(2) an integer, call it "left", that tells the leftmost cell of the part of the array to be shifted;

(3) an integer, call it "right", that tells the rightmost cell of the part of the array to be shifted;

(4) a positive integer, call it "distance" that tells how many cells to shift by.

The function should make sure that left is less than or equal to right, and that distance is greater

than zero. If either of these conditions fails, the function should return the value 1 to indicate an error.

Otherwise it should shift by distance cells the contents of the array cells with subscripts running from

left to right . Thus, for example, if the array passed to the function looks like this:

0 1 2 3 4 5 6 7 8 9 10 ....

5.8 | 2.6 | 9.1 | 3.4 | 7.0 | 5.1 | 8.8 | 0.3 | -4.1 | 8.0 | 2.7 | etc.

and if left has the value 3 , right has the value 7 , and distance has the value 2 , then the

function should shift the contents of cells 3 , 4 , 5 , 6 , and 7 to the right by 2 cells, so that when the function

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

0 1 2 3 4 5 6 7 8 9 10 ....

5.8 | 2.6 | 9.1 | ??? | ??? | 3.4 | 7.0 | 5.1 | 8.8 | 0.3 | 2.7 | etc.

The question marks in cells 3 and 4 indicate that we don't care what numbers are in those cells when the

function returns. Note that the contents of cells 8 and 9 have changed, but the contents of cell 10 is

unchanged. The function need not take any precautions against the possibility that the cells will be shifted

beyond the end of the array (the calling function should be careful not to let that happen).

**35.** 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.

**36.** Write a function named "concatenate" that copies the cells of one array into a larger array, and then

copies the cells of another array into the larger array just beyond the contents of the first array. The contents of

the cells will be integers. The arguments will be as follows:

(1) the first array that will be copied;

(2) the number of cells that will be copied from the first array;

(3) the second array that will be copied;

(4) the number of cells that will be copied from the second array;

(5) the large array into which all copying will be performed;

(6) the number of cells available in the large array.

If the function discovers that the number of cells in the large array is not large enough to hold all the numbers to

be copied into it, then the function should return 0 to indicate failure. Otherwise it should return 1 . The

function should not alter the contents of the first two arrays. To take an example, if the first two arrays passed

to the function look like this:

0 1 2 3 4 5 6 0 1 2 3

58 | 26 | 91 | 34 | 70 | 34 | 88 and 29 | 41 | 10 | 66

then, provided the size of the large array is at least 11, the large array should look like this when the function

returns:

0 1 2 3 4 5 6 7 8 9 10

58 | 26 | 91 | 34 | 70 | 34 | 88 | 29 | 41 | 10 | 66

**37.** Write a function named "number\_of\_matches" that compares the initial parts of two character

arrays to see how many pairs of cells match before a difference occurs. For example, if the arrays are

0 1 2 3 4 5 0 1 2 3 4

**Error!** and **Error!**

then the function should return the value 3 because only the first three pairs of cells in the arrays match (cell 0

matches cell 0, cell 1 matches cell 1, and cell 2 matches cell 2). Each of the character arrays will end with the

character whose ASCII value is zero; this character, called NUL, is denoted by '\0' in the C and C++

programming languages (see the two arrays shown above). The pairwise cell comparisons should not go

beyond the end of either array. If the two arrays are identical all the way to their terminating NUL characters,

return the number of non-NUL characters. The function should take only two parameters, namely the two

character arrays to be compared.

**38.** 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.

19

**39.** Write an entire C++ program that reads a positive integer entered by an interactive user and then prints out

all the positive divisors of that integer in a column and in decreasing order. The program should allow the user

to repeat this process as many times as the user likes. Initially, the program should inform the user about how

the program will behave. Then the program should prompt the user for each integer that the user wishes to

enter.

The program may be terminated in any of two ways. One way is to have the program halt if the user

enters an integer that's negative or zero. In this case the user should be reminded with each prompt that the

program can be terminated in that way. Alternatively, after an integer has been entered and the divisors have

been printed, the program can ask the user whether he/she wishes to enter another integer. In this case, when

the user accidentally enters a zero or negative integer to have its divisors calculated, the program should inform

the user that the input is unacceptable and should allow the user to try again (and again!).

Here is an illustration of how the program and the interactive user might interact. The user's responses

to the program are shown in bold italics.

This program is designed to exhibit the positive divisors of

positive integers supplied by you. The program will repeatedly

prompt you to enter a positive integer. Each time you enter a

positive integer, the program will print all the divisors of

your

integer in a column and in decreasing order.

Please enter a positive integer: ***36***

36

18

12

9

6

4

3

2

1

Would you like to see the divisors of another integer (Y/N)? ***y***

Please enter a positive integer: ***-44***

-44 is not a positive integer.

Please enter a positive integer: ***0***

0 is not a positive integer.

Please enter a positive integer: ***109***

109

1

Would you like to see the divisors of another integer (Y/N)? ***m***

Please respond with Y (or y) for yes and N (or n) for no.

20

Would you like to see the divisors of another integer (Y/N)? ***n***

PART 3:

1. Write a program that uses printf() to print all the numbers from 1 to 100,

with two exceptions. For numbers divisible by 3, print "Fizz" instead of the

number, and for numbers divisible by 5 (and not 3), print "Buzz" instead.

When you have that working, modify your program to print "FizzBuzz" for

numbers that are divisible by both 3 and 5 (and still print "Fizz" or "Buzz"

for numbers divisible by only one of those).

(This is actually an interview question that has been claimed to weed out

a significant percentage of programmer candidates. So if you solved it, your

labor market value just went up.)

***2.* Minimum**

Write a functionmin that takes two arguments and returns their minimum and another that returns their max maximum

**Recursion**

We’ve seen that % (the remainder operator) can be used to test whether a

number is even or odd by using % 2 to see whether it’s divisible by two. Here’s

another way to define whether a positive whole number is even or odd:

55

• Zero is even.

• One is odd.

• For any other number *N*, its evenness is the same as *N* - 2.

Define a recursive function isEven corresponding to this description. The

function should accept a single parameter (a positive, whole number) and return

a Boolean.

Test it on 50 and 75. See how it behaves on -1. Why? Can you think of a

way to fix this?

**Bean counting**

You can get the Nth character, or letter, from a string by writing string\_name[n].

The returned value will be a string containing only one character (for example,

"b"). The first character has position 0, which causes the last one to be found at

position string.length - 1. In other words, a two-character string has length

and its characters have positions 0 and 1.

Write a function countBs that takes a string as its only argument and returns

a number that indicates how many uppercase “B” characters there are in the

string.

Next, write a function called countChar that behaves like countBs, except

it takes a second argument that indicates the character that is to be counted

(rather than counting only uppercase “B” characters). Rewrite countBs to

make use of this new function.

56

*“On two occasions I have been asked, ‘Pray, Mr. Babbage, if you put*

*into the machine wrong figures, will the right answers come out?’*

*[...] I am not able rightly to apprehend the kind of confusion of ideas*

*that could provoke such a question.”*

—Charles Babbage, Passages from the Life of a Philosopher (1864)

write two functions, reverseArray and reverseArrayInPlace. The first, reverseArray, takes an array as argument and produces a *new* array that has the same elements in the inverse order