1.（Arithmetic, Smallest and Largest）Write a program that inputs three integers from the keyboard and prints the sum, average, product, smallest, and largest of these numbers. The screen dialog should appear as follows:

Input three different integers: 13 27 14

Sum is 54

Average is 18

Product is 4914

Smallest is 13

Largest is 27

2. (Car-Pool Savings Calculator) Research several car-pooling websites. Create an application that calculates your daily diving cost, so that you can estimate how much money could be saved by car pooling, which also has other advantage such as reducing carbon emission and reducing traffic congestion. The application should input the following information and display the user’s cost per day of driving to work:

a) Total miles driven per day.

b) Cost per gallon of gasoline.

c) Average miles per gallon.

d) Parking fees per day.

e) Tolls per day.

3. (Date Class) Create a class called *Date* that includes three pieces of information as data members: a month (type int), a day (type int) and a year (type int). Your class should have a constructor with three parameters that uses the parameters to initialize the three data members. For the purpose of this exercise, assume that the values provided for the year and day are correct, but ensure that the month value is in the range 1-12; if it is not, set the month to 1. Provide a set and a get function for each data member. Provide a member function *displayDate* that displays the month, day and year separated by forward slashes (/). Write a test program that demonstrates class *Date*'s capabilities.

4. (Account Class) Create a class called *Account* that a bank might use to represent customers' bank accounts. Your class should include one data member of type *int* to represent the account balance. Provide a constructor that receives an initial balance and uses it to initialize the data member. The constructor should validate the initial balance to ensure that it is greater than or equal to 0. If not, the balance should be set to 0 and the constructor should display an error message indicating that the initial balance was invalid. The class should provide three member functions. Member function *credit* should add an amount to the current balance. Member function *debit* should withdraw money from the Account and should ensure that the debit amount does not exceed the Account's balance. If it does, the balance should be left unchanged and the function should print a message indicating “Debit amount exceeded account balance”. Member function *getBalance* should return the current balance. Create a program that creates two *Account* objects and tests the member functions of class *Account*.

4. (Date Class) Create a class called *Date* that includes three pieces of information as data members: a month (type int), a day (type int) and a year (type int). Your class should have a constructor with three parameters that uses the parameters to initialize the three data members. For the purpose of this exercise, assume that the values provided for the year and day are correct, but ensure that the month value is in the range 1-12; if it is not, set the month to 1. Provide a set and a get function for each data member. Provide a member function *displayDate* that displays the month, day and year separated by forward slashes (/). Write a test program that demonstrates class *Date*'s capabilities.

5. (Bubble Sort) In the bubble sort algorithm, smaller values gradually “bubble” their way upward to the top of the array like air bubbles rising in water, while the larger values sink to the bottom. The bubble sort makes several passes through the array. On each pass, successive pairs of elements are compared. If a pair is in increasing order (or the values are identical), we leave the values as they are. If a pair is in decreasing order, their values are swapped in the array. Write a program that sorts an array of 10 integers using bubble sort.

6. Develop a C++ program that will determine whether a department-store customer has exceeded the credit limit on a charge account. For each customer, the following facts are available:

a) Account number (an integer)

b) Balance at the beginning of the month

c) Total of all items charged by this customer this month

d) Total of all credits applied to this customer's account this month

e) Allowed credit limit

The program should use a while statement to input each of these facts, calculate the new balance (= beginning balance + charges credits) and determine whether the new balance exceeds the customer's credit limit. For those customers whose credit limit is exceeded, the program should display the customer's account number, credit limit, new balance and the message “Credit Limit Exceeded.”

Enter account number (-1 to end): 100

Enter beginning balance: 5394.78

Enter total charges: 1000.00

Enter total credits: 500.00

Enter credit limit: 5500.00

New balance is 5894.78

Account: 100

Credit limit: 5500.00

Balance: 5894.78

Credit Limit Exceeded.

Enter Account Number (or -1 to quit): 200

Enter beginning balance: 1000.00

Enter total charges: 123.45

Enter total credits: 321.00

Enter credit limit: 1500.00

New balance is 802.45

Enter Account Number (or -1 to quit): -1

7. (Function Template maximum) Write a program that uses a function template called maximum to determine the largest of three arguments. Test the program using integer, character and floating-point number arguments.

8. (Computer-Assisted Instruction) The use of computers in education is referred to aas computer-assisted instruction (CAI). Write a program that will help an elementary school student learn multiplication. Use the rand function to produce two positive one-digit integers. The program should then prompt the user with a question, such as

*How much is 6 times 7?*

The student then inputs the answer. Next, the program checks the student’s answer. If it’s correct, display the message “Very good!” and ask another multiplication question. If the answer is wrong, display the message “No, Please try again.” And let the student try the same question repeatedly until the student finally feus it right. A separate function should be used to generate each new question, This function should be called once when the application begins execution and each time the user answers the question correctly.

9. (Pass-by-Value vs. Pass-by-Reference) Write a complete C++ program with two alternate functions specified below, each of which simply triples the variable *count* defined in *main* function. Then compare and contrast the two approaches. These two functions are

a) function tripleByValue that passes a copy of *count* by value, triples the copy and returns the new value and

b) function tripleByReference that passes *count* by reference parameter via a reference parameter and triples the original value of *count* through its alias (i. e., the reference parameter).

10. (Selection Sort) **A selection sort** searches an array looking for the smallest element in the array. Then, the smallest element is swapped with the first element of the array. The process is repeated for the subarray beginning with the second element of the array. Each pass of the array results in one element being placed in its proper location. For an array of n elements, n - 1 passes must be made, and for each subarray, n - 1 comparisons must be made to find the smallest value. When the subarray being processed contains one element, the array is sorted. Write recursive method *selectionSort*to perform this algorithm.

11. (Dice Rolling) Write a program that simulates the rolling of two dice. The program should use the *rand* function to roll the first die, and should use *rand* again to roll the second die. The sum of the two values should then be calculates. [Note: Each die can show an integer value from 1 to 6, so the sum pf the two values will vary from 2 to 12, with 7 being the most frequent sum and 2 and 12 being the least frequent sums.] Figure 1 shows the 36 possible combinations of the two dice. Your program should roll the two dice 36,000 times. Print the results in a tabular form. Use a one-dimensional array to tally the numbers of times each possible sum appears. Print the results in a tabular format. Also, determine if the totals are reasonable. (i. e., there are six ways to roll a 7, so approximately one-sixth of all the rolls should be 7)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| 6 | 7 | 8 | 9 | 10 | 11 | 12 |

12. An integer is said to be prime if it is divisible only by 1 and itself. For example, 2, 3, 5 and 7 are prime, but 4, 6, 8 and 9 are not.

1. Write a method that determines if a number is prime.
2. Use this method in an applet that determines and prints all the prime numbers between 1 and 10,000. How many of these numbers do you really have to test before being sure that you have found all the primes?
3. Initially you might think that n/2 is the upper limit for which you must test to see if a number is prime, but you need only go as high as the square root of n. Why? Rewrite the program and run it both ways. Estimate the performance improvement.

13. Airline Reservations System) A small airline has just purchased a computer for its new automated reservations system. You have been asked to program the new system. You are to write a program to assign seats on each flight of the airline’s only plane (capacity: 10 seats).

Your program should display the following menu of alternatives:

*Please type 1 for "smoking"  
Please type 2 for "nonsmoking"*

If the person types 1, your program should assign a seat in the smoking section (seats 1-5). If the person types 2, your program should assign a seat in the nonsmoking section (seats 6-10). Your program should then print a boarding pass indicating the person’s seat number and whether it is in the smoking or nonsmoking section of the plane.

Use a single-subscripted array to represent the seating chart of the plane. Initialize all the elements of the array to 0 to indicate that all seats are empty. As each seat is assigned, set the corresponding elements of the array to 1 to indicate that the seat is no longer available.

Your program should, of course, never assign a seat that has already been assigned. When the smoking section is full, your program should ask the person if it is acceptable to be placed in the nonsmoking section (and vice versa). If yes, make the appropriate seat assignment. If no, print the message "Next flight leaves in 3 hours."