E.S MARIE ADELAIDE

Computer science department

S6MCE

EXERCISES TO BE DONE IN COVID-19 PERIOD (STAY@ HOME)

I. C++ PROGRAMMING & FUNCTION

1. How many different types can the elements of an array have?
2. What type and range must an array's subscript have?
3. What values will the elements of an array have when it is declared if it does not include an initializer?
4. What values will the elements of an array have when it is declared if it has an initializer with fewer values than the number of elements in the array?
5. What happens if an array's initializer has more values than the size of the array?
6. How does an **enum** statement differ from a **typedef** statement?
7. When a multi-dimensional array is passed to a function, why does C++ require all but the first dimension to be specified in the parameter list?
8. Write and test the following function that returns the minimum value among the first **n** elements of the given array: **float min(float a[],int n);**
9. Write and test the following function:
   void append(int a[],int m,int b[],int n);
   The function appends the first **n** elements of the array **b** onto the end of the first **m** elements of the array **a**. It assumes that **a** has room for at least **m + n** elements. For example, if **a** is {22,33,44,55,66,77,88,99} and **b** is {20,30,40,50,60,70,80,90} then the call
   append(a,5,b,3) would transform **a** into {22,33,44,55,66,20,30,40}. Note that **b** is left unchanged and only **n** elements of **a** are changed.
10. Write and test the function
    void insert(float a[],int& n,float x)
    This function inserts the item **x** into the sorted array **a** of **n** elements and increments **n**. The new item is inserted at the location that maintains the sorted order of the array. This requires shifting elements forward to make room for the new **x**. (Note that this requires the array to have at least **n**+1 elements allocated.)
11. Implement the **Insertion Sort** algorithm for sorting an array of n elements. In this algorithm, the main loop index i runs from 1 to **n**–1. On the ith iteration, the element a[i] is "inserted" into its correct position among the subarray a[0..i]. This is done by shifting one position up all the elements in the subarray that are greater than a[i]. Then a[i] is copied into the gap between the elements that are less than or equal to a[i] and those that are greater. (Hint: use the insert() algorithm from YOUR BOOK S5MCE.)
12. Implement the **Selection Sort** algorithm for sorting an array of n elements. This algorithm has **n**–1 iterations, each selecting the next largest element a[j] and swapping it with the

element that is in the position where a[j] should be. So on the first iteration it selects the largest of all the elements and swaps it with a[n-1], and on the second iteration it selects the
largest from the remaining unsorted elements a[0..n-2] and swaps it with a[n-2], **etc**. On its **i**th iteration it selects the largest from the remaining unsorted elements a[0..n-i] and swaps it with a[n-i]. (Hint: use the same loops as WELL)

13. Rewrite and test the Bubble Sort function presented in PREVIOUS QUESTION, as an **indirect sort**. Instead of moving the actual elements of the array, sort an index array instead

II. POINTERS (A)

1. How do you access the memory address of a variable?
2. How do you access the contents of the memory location whose address is stored in a pointer
variable?
3. Explain the difference between the following two declarations:
int n1=n;
int& n2=n;

4. Explain the difference between the following two uses of the reference operator &:
int& r = n;
p = &n;

5. Explain the difference between the following two uses of the indirection operator *:
int* q = p;
n = *p;
6. True or false? Explain:

a. If (x == y) then (&x == &y).
b. If (x == y) then (*x == *y).

7.
a. What is a “dangling pointer”?

b. What dire consequences could result from dereferencing a dangling pointer?
c. How can these dire consequences be avoided?

8. What is wrong with the following code:
int& r = 22;
9. What is wrong with the following code:
int* p = &44;
10. What is wrong with the following code:
char c = 'w';
char p = &c;
11. Why couldn’t the variable **ppn** in Example 7.6 on page 160 be declared like this:
int ppn=&&n;**
12. What is the difference between “static binding” and “dynamic binding”?

13. What is wrong with the following code:
    char c = 'w';
    char* p = c;
14. What is wrong with the following code:
    short a[32];
    for (int i = 0; i < 32; i++)
    ***a++ = i*i;**
15. Determine the value of each of the indicated variables after the following code executes. Assume that each integer occupies 4 bytes and that **m** is stored in memory starting at byte **0x3fffd00**.
    int m = 44;
    int* p = &m;
    int& r = m;
    int n = (*p)++;
    int* q = p - 1;
    r = *(--p) + 1;
    ++*q;
    a. **m**
    b. **n**
    c. **&m**
    d. ***p**
    e. **r**
    f. ***q**
16. Classify each of the following as a mutable lvalue, an immutable lvalue, or a non-lvalue:
    a. double x = 1.23;
    b. 4.56*x + 7.89
    c. const double Y = 1.23;
    d. **double a[8] = {0.0};**
    e. **a[5]**
    f. **double f() { return 1.23; }**
    g. **f(1.23)**
    h. **double& r = x;**
    i. **double* p = &x;**
    j. ***p**
    k. **const double* p = &x;**
    l. **double* const p = &x;**

17. What is wrong with the following code:
    float x = 3.14159;
    float* p = &x;
    short d = 44;
    short* q = &d;
    p = q;

18. What is wrong with the following code:
    int* p = new int;
    int* q = new int;
    cout << "p = " << p << ", p + q = " << p + q << endl;
19. What is the only thing that you should ever do with the **NULL** pointer?
20. In the following declaration, explain what type **p** is, and describe how it might be used:
    double** p;**
21. If **x** has the address **0x3fffd1c**, then what will values of **p** and **q** be for each of the following:
    double x = 1.01;
    double* p = &x;
    double* q = p + 5;
22. If **p** and **q** are pointers to **int** and **n** is an **int**, which of the following are legal:
    a. **p + q**
    b. **p – q**
    c. **p + n**
    d. **p – n**
    e. **n + p**
    f. **n – q**
23. What does it mean to say that an array is really a constant pointer?
24. How is it possible that a function can access every element of an array when it is passed only
    the address of the first element?
25. Explain why the following three conditions are true for an array **a** and an int **i**:
    a[i] == *(a + i);
    ***(a + i) == i[a];**
    a[i] == i[a];
26. Explain the difference between the following two declarations:
    double * f();
    double (* f)();
27. Write a declaration for each of the following:
    a. an array of 8 floats;
    b. an array of 8 pointers to float;
    c. a pointer to an array of 8 floats;
    d. a pointer to an array of 8 pointers to float;
    e. **a function that returns a** float**;**
    f. **a function that returns a pointer to a** float**;**
    g. **a pointer to a function that returns a** float**;**
    h. **a pointer to a function that returns a pointer to a** float**;**

III. POINTER(B)

1. Write a function that uses pointers to copy an array of double.

2. Write a function that uses pointers to search for the address of a given integer in a given array. If the given integer is found, the function returns its address; otherwise it returns NULL.
3. Write a function that is passed an array of n pointers to floats and returns a newly created array that contains those n float values.

IV. STANDARD C++ STRING

Write a program that counts and prints the number of lines, words, and letter frequencies in its input. For example, the input:
Two roads diverged in a yellow wood,
And sorry I could not travel both
And be one traveler,long I stood
And looked down one as far as I could
To where it bent in the undergrowth;
would produce the output:
The input had 5 lines,37 words,
and the following letter frequencies:
A: 10 B: 3 C: 2 D: 13 E: 15 F: 1 G: 3 H: 4
I: 7 J: 0 K: 1 L: 8 M: 0 N: 12 O: 20 P: 0
Q: 0 R: 11 S: 5 T: 11 U: 3 V: 3 W: 6 X: 0
Y: 2 Z: 0

V. VISUALBASIC DOT NET(2015)

1. After writing all steps used to make install file (.exe),write vb .net code for BMI application .

VI. JAVA PROGRAMMING

1. Write java programming language that will display the sum of two integers from the user and the after displays the answer in **Dialogbox**