POINTER VARIABLES

Short Answer

1. * Distinguish between a *pointer to a constant* and a *constant pointer*.

   Write the definitions of `ptr1`, which is a *pointer* to a *constant integer*, and `ptr2`, which is a *constant pointer* to an integer.

2. * What is a *null pointer*? What is a *void pointer*?

3. * What is the purpose of the `new/delete` operator?

4. * Consider the following C++ statement, `int count = 10;`

   Write a statement that displays the address of the variable `count`.

   Write a statement that displays the content held by the variable `count`.

5. * Write C++ statements to complete the following requirements consecutively

   - Define a double pointer variable, `dblPtr`, which is a pointer to a double value.
   - Dynamically allocates memory for the pointer variable `dblPtr` created above.
   - Release the dynamic memory that was requested for `dblPtr`.

6. ** What are the advantages of declaring a pointer parameter as a *constant pointer*?

7. ** Distinguish these two following C++ statements, `++*ptr;` and `*ptr++;`

8. ** Suppose you want to copy the content held in the array `b[10]` to the array `a[10]`.

   Could it possibly be done with the following C++ statement, `b = a;` ? Justify your answer.

9. ** Let `ptr` be a pointer to an integer and `ptr` holds the address <120>. On a system with 4-byte integers, what address will be in `ptr` after the following C++ statement, `ptr += 10;` ?

10. ** Consider the following C++ statements,

```
char arr[] = "Surprised";
const char *ptr = "Surprised";
```

   Would `arr[3]` and `ptr[3]` fetch the same character? Justify your answer.

11. ** What is the output of the following C++ code fragment?

```
int arr[10][20];
cout << sizeof(arr[4]);
```

12. ** Write a C++ function called `exchange` that takes two integer pointer variables and exchanges the values in those variables.

13. ** Consider the following C++ definition of an array, `int arr[2] = {0, 1, 2, 3};`

   Point out the (syntax/logic/semantic) errors if there is any.

14.** Consider the following C++ code fragment. Identify the syntax/logic/semantic errors in the code fragment if there is any. Suggest the correction for each of the errors.

```
int *findMax(int *a, int *b) {
      int max;
      if(*a > *b)
            max=*a;
      else
            max = *b;
      Return &max;
}
int main() {
      int x=7, y=15, max;
      max = findMax(&x, &y);
      return 0;
}
```

15.** Consider the following C++ definition, `const int numbers[SIZE] = { 18, 17, 12, 14 };` Suppose you pass `numbers` to the function `processArray` by using the following statement, `processArray(numbers, SIZE);` Write the correct C++ function header for `processArray`.

16.** Consider the following C++ code fragment.

```
int x = 7;
int *iptr = &x;
```

What will be displayed if you send the expression `*iptr` to `cout`?

What happens if you send the expression `iptr` to `cout`?

17.** What is the output of the following C++ code fragment? Note that the notation *nullptr* here refers to the null pointer, while it can be represented differently in various IDEs.

```
int x = 50, y = 60, z = 70;
int *ptr = nullptr;
cout << x << " " << y << " " << z << endl;
ptr = &x;
*ptr *= 10;
ptr = &y;
*ptr *= 5;
ptr = &z;
*ptr *= 2;
cout << x << " " << y << " " << z << endl;
```

18.** Will the following C++ program compile without error? If yes, what will the output of the following program be? If no, point out the error(s).

```
int foo(int i) { return i; }
int main() {
      int a = 2;
      int *b = &a;
      cout << foo(*b) << endl;
      return 0;
}
```

19.* Given `arr` is an array of integers. Will each of the below code segments show "True" or "False"?

A	`if (arr < &arr[1])`
	`  cout << "True";`
	`else`
	`  cout << "False";`
		C	`if (arr != &arr[2])`
			`  cout << "True";`
			`else`
			`  cout << "False";`

B	`if (&arr[4] < &arr[1])`
	`  cout << "True";`
	`else`
	`  cout << "False";`
		D	`if (arr != &arr[0])`
			`  cout << "True";`
			`else`
			`  cout << "False";`

20.* Is each of the following definitions valid or invalid? If any is invalid, why?

A	`int ivar;` `int *iptr = &ivar;`	D	`float fvar;` `int *iptr = &fvar;`
B	`int ivar, *iptr = &ivar;`	E	`int nums[50], *iptr = nums;`
C	`int *iptr = &ivar;` `int ivar;`		

Fill-in-the-Blank

1. * Fill in each of the following blanks with an appropriate terminology.
   - A _____ is a special variable that holds a memory address.
   - The _____ operator gives you the address of a variable while the _____ operator gets you the value from the address.

2. ** How many bytes of memory are allocated for the following variables? Assume that a character takes 1 byte, an integer takes 4 bytes, a floating-point number takes 4 bytes, and a pointer takes 8 bytes.
   - `float a[] = {4.75};` _____
   - `char b[] = “4.75”;` _____
   - `const char *b = “4.75”;` _____
   - `char animals[][10] = {“lion”, “elephant”, “tiger”, “cat”};` _____
   - `int c[10] = {1,2,3};` _____

3. ** Consider the following C++ code fragment.

```
int a[] = {1, 2, 3, 4};
*(a+2) += 3;
```

   The value of `a[2]` after the fragment is executed is _____.

4. ** Consider the following C++ code fragment.

```
int a[] = {1, 2, 3, 4};
int* p = a + 1;
p++; (*p)++;
```

   The value of `p` after the fragment is executed is _____.

5. ** Consider the following C++ statements.

```
double value = 29.7;
double *ptr = &value;
```

   Write a `cout` statement that uses `ptr` to display the content of `value`.

```
cout << ______________________ ;
```

6. ** Consider the following C++ statements.

```
int set[10];
```

   Write a `cin` statement that uses pointer notation to assign the value inputted to `set[7]`.

```
cin >> ______________________;
```

7. ** Rewrite the following loop using pointer notation (with the indirection operator) instead of subscript notation.

`for (int i = 0; i < 100; ++i)` `  cout << arr[i] << endl;`	

8. ** Look at the following C++ function definition. In this function, n is a reference variable. Rewrite the function so that n is a pointer.

`void getNumber(int &n){`
`  cout << "Enter a number: ";`
`  cin >> n;`
`}`	

9. ** Consider the following C++ program.

```
void change(int* ptr){
      *ptr = 25;
}
int main(){
      int num = 10;
      change(___________)
      return 0;
}
```

Write a statement that calls the function `change` with the integer variable `num`.

The value of `num` after the statement is executed is _____ .

10.** Given two integer variables, a and b, which are defined in the C++ main function.

How do you define the function `swap` for each of the following main functions such that the values stored in a and b will be swapped?

`int main() {`
`  int a = 45, b = 35;`
`  swap(&a, &b);`
`}`
	`void swap(_____, _____) {`
	
	`}`

`int main() {`
`  int a = 45, b = 35;`
`  swap(a, b);`
`}`
	`void swap(_____, _____) {`
	
	`}`

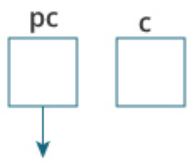
11.** Consider the following C++ function

```
int mystery(int a, int* b, int& c) {
      a++;
      (*b)++;
      c++;
      return a;
}
```

Assume that the variables `a,b,c` and `d` are all initialized to `0`. Identify their values after each of the following function calls.

	a	b	c	d
`mystery(a, &b, c);`				
`mystery(c, &d, a);`				
`c = mystery(b, &a, d);`				
`a = mystery(a, &a, a);`				

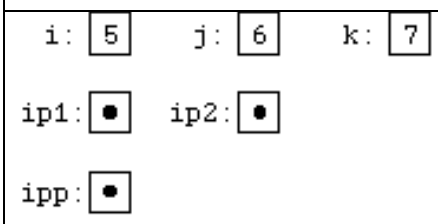
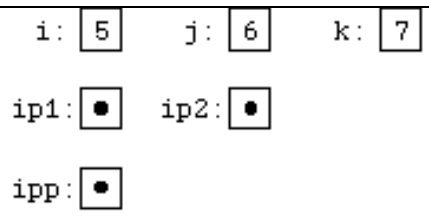
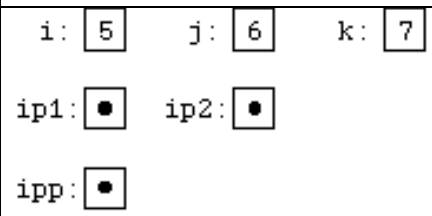
12.** In the following table, the first row shows a sequence of C++ statements (from left to right), while the second row shows corresponding demonstrations for each of the above statement. For example, because the variables, `pc` and `c`, in the first statement are not initialized, the pointer `pc` points to no specific address, and variable `c` has an address but contains a random garbage value and hence is shown empty. Provide the demonstrations for subsequent statements.

`int* pc, c;`	`c = 22;`	`pc = &c;`	`c = 11;`	`*pc = 2;`
				

13.*** Consider the following C++ statements,

```
int i = 5, j = 6, k = 7;
int *ip1 = &i, *ip2 = &j;
int **ipp;
```

Illustrate each of the following situations with the box-and-arrow notation

`ipp = &ip1;`	`ipp = &ip1;`	`*ipp = &k;`
		

14.*** Arrays are similar to pointers. One difference is that we cannot change the value of the array address, but we can change the value of a pointer variable. Using the C++ declaration given below, write down the result for each of the valid statements or "INVALID" for each of the invalid statements. Briefly give your reasons.

```
int i[15], *p1;
float x[15], *p2;
```

- `x = x + 1;` _____
- `x = p2 + 1;` _____
- `p2 = x + 1;` _____
- `p2 = p2 + 1;` _____
- `x[2] = x[2] + 1;` _____
- `&x[2] = p2;` _____

15.* Assume the lines are executed sequentially on a system with 4-byte integers and the address of the blocks array is 4434. Give the value of the left-hand side variable in each assignment.

C++ program	Left-hand side variable

`int main(){`
`  char blocks[3] = {'A','B','C'};`
`  char *ptr = &blocks[0];`
`  char temp;`
`  temp = blocks[0];`
`  temp = *(blocks + 2);`
`  temp = *(ptr + 1);`
`  temp = *ptr;`
`  ptr = blocks + 1;`
`  temp = *ptr;`
`  temp = *(ptr + 1);`
`  ptr = blocks;`
`  temp = *++ptr;`
`  temp = ++*ptr;`
`  temp = *ptr++;`
`  temp = *ptr;`
`  return 0;`
`}`	

True or False

Choose T (True) or F (False) for each of the following statements and then briefly explain in one or two sentences.

1. T F Arrays allow random access.
2. T F Arrays always occupy contiguous memory area.
3. T F When the indirection operator is used with a pointer variable, you are working with the value the pointer is pointing to.
4. T F An integer variable memory location can be modified only through `int*`
5. T F On a system with 4-byte integers, the next address of an int pointer will be 4 bytes apart.
6. T F A pointer variable that has not been initialized is called a *null pointer*.
7. T F `void*` is not a valid pointer.
8. T F For an int pointer `p`, `p[10]` is a valid expression.
9. T F Uninitialized memory accesses may lead to program crash.
10. T F For a char pointer `p`, `p--` is an invalid expression.
11. T F If `p` is a pointer variable, then the statement `p = p * 2;` is valid in C++.
12. T F Pointers enable programs to simulate call by value.
13. T F The following statement defines two integer pointers, `int *aptr, bptr;`
14. T F The & and * operators are complements of one another—when they are both applied consecutively to a pointer, in either order, there is no modification.
15. T F `delete ptr[];` is the right syntax to de-allocate a single dimensional array of integer pointer dynamically.

Algorithm Workbench

1. * Consider the following statement, `char s[] = "hello world";` Write code to make a copy of the string in `s` and have it referred to by the variable `p`.
2. * Write code to input in a non-pointer variable and display the same value using pointer.
3. * Write code to enter integers and store them in the variables, `iA` and `iB`, respectively. There are also two integer pointers, i.e. `ptrA` and `ptrB`. Assign the value of `iA` to `ptrA` and the value of `iB` to `ptrB`. Display the content of `ptrA` and `ptrB`.

For the following C++ code segment, use the pointer notation ONLY. Do NOT use the array index [ ].

4. ** Given the string, `"A string"`, and a pointer `ptr` that points to that string. Use `ptr` to print on one line the letter at the index `0`, the pointer position and the letter `t`. Update the pointer to pointer + 2. Then, in another line, use `ptr` to print the whole content of the pointer and the letters `r` and `g` of the string.
5. ** Write a piece of code that prints the characters in a C-string in reverse order.

```
char s[10] = "abcde";
char* cptr;
// WRITE YOUR CODE HERE
```

6. ** Write the function, `countEven(int*, int)`, to receive an integer array and its size, and then return the number of even numbers in the array.
7. ** Write a function to return a pointer to the maximum value of an array of double numbers. If the array is empty, return NULL. The prototype of the function is as follows.

```
double* maximum(double* a, int size);
```

8. ** Write the function, `myStrLen(char*)`, to return the length of the parameter C-string, yet without using the C-string function `strlen`.
9. ** Write the function, `myStrContains(char*, char)`, that returns true if the 1st parameter C-string contains the 2nd parameter character, or false otherwise, without using the C-string search functions like `strchr` or `strrchr`.
10. ** Write the function, `revString(char*)`, to receive the parameter C-string. The function returns nothing. You may use the C-string handling functions if you wish.

```
void revString(char* ptr){

   // WRITE YOUR CODE HERE

}
```

```
int main(){
  char s[10] = "abcde";
  revString(s);  // call the function
  return 0;
}
```