CHARACTERS and STRINGS

In this material, we assume that all necessary header files are included for every code segment.

Short Answer

1. * Which header file must be included in a program using *C-string* conversion functions, such as `atof` and `itoa`? Which header file must be included in a program using *C++ string* property functions, such as `empty` and `length`?

2. * Which function is used to locate a substring in a given *C-string*?
Which function is used to locate a substring in a given *C++ string*?

3. * What are the advantages and disadvantages of *C++ strings* over *C strings*?

4. * How do you make an array of C-strings? How do you make an array of *C++ strings*?

5. ** Given a pair of *C-strings*, `cstr1` and `cstr2`, which are already initialized. Is it possible to use relational operators (i.e., ==, !=, >, <, <=, >=) on these strings? For each case, write an example to use that operator if it is possible; otherwise, suggest some alternative built-in functions.
Repeat the question for a pair of *C++ strings*, `str1` and `str2`.

6. ** What is the main difference between the `cin` statement and the `getline` function?

7. ** What does each of the following code segments output? Briefly explain each case.

-
  ```
  char a[][20] = { "Argentina", "Korea", "Greece", "Nigeria"};
  cout << *(a+1) << *a[0] << a[3] << a[3][1] << endl;
  ```

-
  ```
  char FRUIT[16];
  strcpy(FRUIT, "Blueberry blackberry!");
  cout << FRUIT << endl;
  ```

-
  ```
  char s[20] = "pencil", p[20];
  int length = strlen(s);
  for (int i = 0; i < length; i++)
        p[i] = s[length - i];
  cout << p;
  ```

-
  ```
  const char *p1 = "India";
  const char *p2 = p1;
  p1 = "Bharat";
  cout << p1 << " " << p2;
  ```

8. ** What does each of the following code segments output?

-
  ```
  string str("steve jobs is legend");
  str.erase(str.begin() + 5, str.end() - 7);
  cout << str << endl;
  ```

-
  ```
  string str ("Microsoft");
  for (int i = 0; i < str.length(); i++)
        cout << str.at(i-1);
  ```

-
  ```
  string str("Geeksfor");
  str.insert(8,"Geeks");
  cout << str << endl;
  ```

-
  ```
  string A = "Good morning";
  string B = "Good afternoon";
  if (A > B)
        cout << A;
  else
        cout << B;
  ```

9. ** Given c is a char variable. What does c hold after each of the following statements executes?

Statement	Contents of c
`c = toupper('a');`	__________
`c = toupper('B');`	__________
`c = tolower('D');`	__________
`c = toupper('e');`	__________

10.** Consider the following declaration of a C-string variable, where SIZE is a defined constant:

```
char ourString[SIZE] = "Hi there!";
```

Write a statement to reassign all positions of ourString the value 'X', leaving the length the same as before.

Explain how the following code can destroy the contents of memory beyond the end of the array.

```
int index = 0;
while (ourString[index] != '\0'){
      ourString[index] = 'X';
      index++;
}
```

Modify the loop to protect against inadvertently changing memory beyond the end of the array.

11.** Point out the syntax errors and/or the semantic error in the following code fragments.

-
  ```
  char string1[] = "Billy",
  char string2[] = " Bob Jones";
  strcat(string1, string2);
  ```

-
  ```
  char numeric[5];
  int x = 123;
  numeric = atoi(x);
  ```

-
  ```
  char str1 = “blueBerry”;
  char str2[] = ‘Blueberry’;
  if (str1 == str2)
        cout << “Equal”
  else
        cout >> “Not equal”;
  ```

12.** Consider the following code segment. What is the output of the code segment?

```
1     char word[] = "terryology";
2     char *cptr = &word[9];
3     while (cptr >= word) {
4           cout << *cptr;
5           cptr--;
6     }
7     cout << endl;
```

Describe the values that are compared in the condition `cptr >= word` in terms of the elements of the `word[]` array. Explain the different meanings of the "*" character on line #2 and line #4.

13.*** What does each of the following code segments output?

-
  ```
  string str = "Alexander Stepanov";
  cout << "str is: " << str << endl;
  cout << "size of str is: " << str.size() << endl;
  str.resize(11);
  cout << "str is: " << str << endl;
  cout << "size of str is: " << str.size() << endl;
  str.resize(20,'.');
  cout << "str is: " << str << endl;
  cout << "size of str is: " << str.size() << endl;
  ```

-
  ```
  string str = "We go step by step to the target";
  cout << "str is: " << str << endl;
  int n = str.find("step");
  string s1 = str.substr(n);
  string s2 = str.substr(n,12);
  cout << "s1 is: " << s1 <<  "and s2 is: " << s2 << endl;
  ```

-
  ```
  string str = "You live only once”
  cout << "size of str is: " << str.size() << endl;
  cout << "length of str is: " << str.length() << endl;
  cout << "max size of str is: " << str.max_size() << endl;
  cout << "capacity of str is: " << str.capacity() << endl;
  ```

14. *** What does each of the following code segments output?

-
  ```
  char name[] = "CoMPutER";
  for (int x = 0; x < strlen(name); x++)
      if (islower(name [x]))
          name [x] = toupper(name[x]);
      else if (isupper(name[x]))
              if (x % 2 == 0)
                  name[x] = tolower(name[x]);
              else
                  name[x] = name[x-1];
  cout << name;
  ```

-
  ```
  void Encrypt(char T[]){
      for (int i = 0; T[i] != '\0'; i += 2)
          if (T[i] == 'A' || T[i] == 'E')
              T[i] = '#';
          else if (islower(T[i]))
                  T[i] = toupper(T[i]);
               else T[i] = '@';
  }
  int main(){
      char text[]="SaVE EArtH";
      Encrypt(text);
      cout << text << endl;
      return 0;
  }
  ```

-
  ```
  int fun(char *p){
      if (p == nullptr || *p == '\0') return 0;
      int current = 1, i = 1;
      while (*(p+current)){
          if (p[current] != p[current-1]){
              p[i] = p[current];
              i++;
          }
          current++;
      }
      *(p+i)='\0';
      return i;
  }
  int main(){
      char str[] = "geeksskeeg";
      fun(str);
      puts(str);
      return 0;
  }
  ```

15. *** How many header files in C/C++ could be used for string manipulation? Differentiate them.

Fill-in-the-Blank

1. * Fill in each of the following blanks with an appropriate terminology.
   - We use ________________ function to locate the last occurrence of a character in a *C-string*.
   - The ________________ character is used to indicate the end of a *C-string.*
   - It takes _____________ characters to store the string “Kate” as a *C-string* and ________________ character to store that string as a *C++ string* object.

2. * Fill in the below table with appropriate operators/built-in functions for *C-strings* and *C++ strings*. Write None if there is no solution

	C strings	*C++ strings*
String assignment		
String comparison		
String concatenation		
Insert a string to an original string at a certain position		
Locate first occurrence of character in string		

3. ** Consider the following statement, `char a[]= “telephone”, b[] = “teleport”;`
   The value returned from the function call, `strcmp(a,b)`, is ___________.

4 . ** Consider the following statement, `char a[]= “home”,  b[]= “work”;`
   What is stored in the string `a` after the function call, `strcat(a,b)`? ________________

5. ** Consider the following statement, `string str = “Harry”;`
   The statement, `cout << str.at(0) << str.at(1);`, will print out ____________

6. ** If a string needs room for 25 characters, how should it be defined so that it will be a valid string?
   ____

7. ** Consider the following C++ string object, `str = “ABCDEFD”.`
   The statement, `cout << str.length( )` will print out ____________________
   The statement, `cout << str.find(“D”)` will print out ____________________
   The statement, `cout << str.rfind(“D”)` will print out ____________________

8. ** Consider the following code segment

```
bool fn(char *s, int len) {
     cout << "len =" << len << endl;
      if (len <=1 )
            return true;
      else
            return( (s[0] == s[len-1]) && fn(s+1, len-2) );
}
```

   What is the return value for the call `fn("abc", 3)`? ______________
   What is printed to cout for the call `fn("noon",4)`? ______________

9. *** What is the result of the following code segment? If `n1` is different from `n2`, briefly explain.

```
char str1[] = "BlueBerry";
char str2[] = { 'B', 'l', 'u', 'e', 'B', 'e', 'r', 'r', 'y' };
int n1 = sizeof(str1) / sizeof(str1[0]);
int n2 = sizeof(str2) / sizeof(str2[0]);
cout << n1 << endl << n2;
```

10. *** Which of the following declarations are equivalent? Give your explanation.

```
a. char stringVar[10] = "Hello";
b. char stringVar[10] = {'H', 'e', 'l', 'l', 'o', '\0'};
c. char stringVar[10] = {'H', 'e', 'l', 'l', 'o'};
d. char stringVar[5] = "Hello";
e. char stringVar[] = "Hello"
```

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 It is possible to use `cin` to input a string that includes whitespace characters.

2. T F It is possible to use `cout` to output a string that includes whitespace characters.

3. T F Relational operators, <, <=, >, >=, ==, !=, are used to compare *C++ string* objects.

4. T F String concatenation on *C-strings* can be done with the operator +, e.g., `c = a+b`

5. T F There is no difference between “123” and 123.

6. T F The `tolower` function toggles the state of a character, i.e. from lowercase to uppercase and vice versa.

7. T F The length of a C++ string can be determined by either `length()` or `size()`.

8. T F If the starting address of a *C-string* is passed into a pointer parameter, it can be assumed that all the characters, from that address up to the byte that holds the null terminator, are part of the string.

9. T F If the two C-strings are not identical, then strcmp() function returns 0.

10. T F If we run the lines

```
string A = "Hello";
string B = A;
B[0] = 'Y';
```

then at the end, the value of `A` is `"Yello"`

Algorithm Workbench

1. * The following `if` statement determines whether `choice` is equal to ‘`Y`’ or ‘`y`’.

    ```
    if (choice == 'Y' || choice == 'y')
    ```

    Simplify the above statement by using either the `toupper` or `tolower` function.

2. * Assume that `str1` and `str2` are C++ string objects. Write code that displays “They are the same!” if the two objects contain the same string.

3. * Consider the following statements. Write a statement that converts the string in `str` to a double and stores the result in `value`.

    ```
    char str[] = "237.89";
    double value;
    ```

For each of the following requirements, try both C-strings and C++ strings if possible

4. ** Write code to get a full name of multiple words and then print it out in reverse order of words.

    Test Data: Donald John Trump

    Expected Output: Trump John Donald

5. ** Write code to receive a string `s` and a non-negative integer `n`. If the number of characters in `s` is larger than `n`, shorten `s` to `n` characters only. Otherwise, `s` is unchanged.

    Test Data: blueberry 5

    Expected Output: blueb

6. ** Write code to receive a string of multiple words and, for every word, capitalize the first letter and decapitalize other letters.

    Test Data: i lOVe YOU

    Expected Output: I Love You

7. ** Write code to check whether a given string is a palindrome (case-insensitive).

    Test Data: nurses run   Expected Output: This is a palindrome.

    Test Data: horses run   Expected Output: This is not a palindrome.

8. ** Write code to receive a string of multiple words and print the set of (distinct) words jn that string, as well as the frequency of each word. Assume that every pair of adjacent words is separated by a single whitespace.

    Test Data: two seats for two people

    Expected Output: two 2 seats 1 for 1 people 1

9. ** Write code to check whether a given string is a pangram (case-insensitive).

    Test Data: the quick brown fox jumps over the lazy dog

    Expected Output: This is a pangram.

    Test Data: I am a student

    Expected Output: This is not a pangram.

10.** A strong password has at least 15 characters of all the following kinds: uppercase letters, lowercase letters, numbers and symbols (such as ` ! " ? $? % ^ & * ( ) _ – + = { [}] : ; @ ' ~ # | < , > . ? Write code to check whether a given password is strong enough.

Test Data: password123	Expected Output: Not strong enough
Test Data: HouseOnSpooner#1500	Expected Output: Strong enough.

11.** Write code to sort an array of strings in descending order of string length. Ties (i.e., strings of equal length) are broken by lexicographical order.

Test Data: one table for two people

Expected Output: people table for one two

12.** Write code that receives a string `s` and then eliminates all leading blanks from the left end of `s` and all trailing blanks from the right end of `s`. It also replaces all subsequent spaces within the string by a single space.

Test Data: _ _ _Hello_ _how_are_ _ you_ _ _ (_ denotes a space, just for easy reading)

Expected Output: Hello how are you

13.** Write code to receive a string and a pattern (which is ensured to be shorter than the string), and it then deletes any but one occurrence of the pattern in the string.

Test Data: two tea to two two two (string) two (pattern)

Expected Output: two tea to

14.*** Write a code segment that uses built-in functions to split a string into multiple substrings following delimiters. For example, "apple and banana" → {"apple", "and", "banana"}.

15.*** Write a function called `swap_heads(s, t)` that returns a single string that consists of `s`, a space, and then `t`, except leading consonants of `s` and `t` have been swapped. If either `s`, or `t`, or both, start with a vowel, then return `s` and `t` with a space between them.

String `s`	String `t`	Expected output
Donald	Trump	Tronald Dump
Bill	Gates	Gill Bates
Emily	Carr	Emily Carr