C String Operations Answers

Sayak Haldar
IIEST, Shibpur

1) Answer is d) void, int and (char *) only

2) Answer is a) Presence of NULL character

3) Answer is b)strrchr(s, c)

Explanation:
char *strchr(const char *s, int c)
this returns a pointer to the first occurrence of a string
char *strrchr(const char *s, int c)
this returns a pointer to the last occurrence of a string

4) Answer is c) strcasecmp(s,t)

5) Answer is a) -1

Explanation: Hello is lexicographically smaller than World. Since, 'H'<'W'

6)Answer is b) o

Explanation: p points to a string “lo”. Now, we increase pointer p by 1.
*(p+1)=p[1]
which is o

7) Answer is b) unequal

8) Answer is d) segmentation fault

Explanation: Since, **str** (character pointer) points to a static memory location (which contains the string “hello”). In case of a character pointer, a string could be written to the memory location pointed by the character pointer only at the time of declaration of str. (i.e. Initialization) We cannot write anything in that location later. Writing a str in the location later causes segmentation fault.

9) Answer is c) Undefined Behaviour

Explanation:

char *strncpy(char *dest, const char *src, size_t n);

The strncpy() function is similar, except that at most n bytes of src are copied. Warning: If there is no null byte among the first n bytes of src, the string placed in dest will not be null-terminated.

If the length of src is less than n, strncpy() writes additional null bytes to dest to ensure that a total of n bytes are written.

Now, here since str1 would clearly not be null terminated, printing it with %s would invoke undefined behaviour

10) Answer is c) 13

11) Answer is a) helloworld 0
Explanation:

char *strcat(char *dest, const char *src);
So, str would act as destination string.h

Now, clearly helloworld 0 would be printed. (why 0 as str[10]? since, str[10]='\0')

However, As I suspect, a warning would be thrown for line 5 saying that incompatible implicit declaration of built-in function ‘strcat’ [enabled by default]
Since, we do not include the string.h header file

12) Answer is b) Always

13) Answer is a) helloworld

Explanation:

char *strncat(char *dest, const char *src, size_t n);
The strncat() function is similar as strcat, except that
* it will use at most n bytes from src; and
* src does not need to be null-terminated if it contains n or more bytes.

As with strcat(), the resulting string in dest is always null-terminated.

Now, as we can see, in case of strncat(str, str1, 9), str acts as destination string,

str1 acts as source string.h
and 9 means at most 9 characters of source string can atmost be concatenated.
So, the whole string would be comfortably concatenated in the destination string.

References:

1) http://www.sanfoundry.com/c-interview-questions-answers/