STRINGS

Solutions

1. Assume sizeof an integer and a pointer is 4 byte. Output?

```
#include <stdio.h>
#define R 10
#define C 20

int main()
{
   int (*p)[R][C];
   printf("%d",  sizeof(*p));
   getchar();
return 0;
}
```

(a) 200 (b) 4
(c) 800 (d) 80

Solution: Option (c)

Explanation:
Output is 10*20*sizeof(int) which is “800″ for compilers with integer size as 4 bytes.

When a pointer is de-referenced using *, it yields type of the object being pointed. In the present case, it is an array of array of integers. So, it prints R*C*sizeof(int).

2. Output of following C program? Assume that all necessary header files are included.

```
int main()
{
  char *s1 = (char *)malloc(50);
  char *s2 = (char *)malloc(50);
  strcpy(s1, "Ravindra");
  strcpy(s2, "Ravula");
  strcat(s1, s2);
  printf("%s", s1);
  return 0;
}
```

(a) RavindraRavula
(b) Ravindra
(c) Ravindra Ravula
(d) Ravula

Solution: Option (a)

Explanation:
strcpy puts \0 at the end.

strcat starts from \0, concatenates string and puts \0 at the end.

3. Consider the following code. The function myStrcat concatenates two strings. It appends all characters of b to end of a. So the expected output is "Ravindra Ravula". The program compiles fine but produces segmentation fault when run.

```
#include <stdio.h>

#include<string,h>

void myStrcat(char *a, char *b)
{
   int m = strlen(a);
   int n = strlen(b);
   int i;
   for (i = 0; i <= n; i++)
      a[m+i]  = b[i];
}

int main()
{
   char *str1 = "Ravindra ";
   char *str2 = "Ravula";
   myStrcat(str1, str2);
   printf("%s ", str1);
   return 0;
}
```

Which of the following changes can correct the program so that it prints "Ravindra Ravula"?

(a) char *str1 = "Ravindra "; can be changed to char str1[100] = "Ravindra ";
(b) char *str1 = "Ravindra "; can be changed to char str1[100] = "Ravindra "; and a line
a[m+n- 1] = '\0′ is added at the end of myStrcat

(c) A line a[m+n-1] = '\0′ is added at the end of myStrcat
(d) A line 'a = (char *)malloc(sizeof(char)*(strlen(a) + strlen(b) + 1)) is added at the beginning of myStrcat()

Solution: Option (a)

4. What is the output of following program?

```
# include <stdio.h>

int main()
{
  char str1[] = "Ravindras";
  char str2[] = {'R', 'a', 'v', 'i', 'n', 'd', 'r', 'a', 's'};
  int n1 = sizeof(str1)/sizeof(str1[0]);
  int n2 = sizeof(str2)/sizeof(str2[0]);
  printf("n1 = %d, n2 = %d", n1, n2);
  return 0;
}
```

(a) n1 = 10, n2 = 9
(b) n1 = 10, n2 = 10
(c) n1 = 9, n2 = 9
(d) n1 = 9, n2 = 10

Solution: Option (a)

Explanation:
The size of str1 is 10 and size of str2 9.

When an array is initialized with string in double quotes, compiler adds a '\0′ at the end.

5. What is the output of following program?

```
#include<stdio.h>

void swap(char *str1, char *str2)
{
 char *temp = str1;
 str1 = str2;
 str2 = temp;
}

int main()
```

```
{
  char *str1 = "Ravindra";
  char *str2 = "ravula";
  swap(str1, str2);
  printf("str1 is %s, str2 is %s", str1, str2);
  return 0;
}
```

(a) str1 is ravula, str2 is Ravindra
(b) str1 is Ravindra, str2 is ravula
(c) str1 is Ravindra, str2 is Ravindra
(d) str1 is ravula, str2 is ravula

Solution: Option (b)

Explanation:
The above swap() function doesn't swap strings. The function just changes local pointer variables and the changes are not reflected outside the function.

6. Predict the output?

```
#include <stdio.h>

int fun(char *str1)
{
  char *str2 = str1;
  while(*++str1);
  return (str1-str2);
}

int main()
{
  char *str = "Ravindrar";
  printf("%d", fun(str));
  return 0;
}
```

(a) 10
(b) 9
(c) 8
(d) Random Number

Solution: Option (b)

Explanation:

The function fun() basically counts number of characters in input string. Inside fun(), pointer str2 is initialized as str1. The statement while(*++str1); increments str1 till '\0' is reached. str1 is incremented by 9. Finally the difference between str2 and str1 is returned which is 9.

7. What does the following fragment of C-program print?

```
char c[] = "GATE2011";
char *p =c;
printf("%s", p + p[3] - p[1]) ;
```

(a) GATE2011
(b) E2011
(c) 2011
(d) 011

Solution: Option (c)

Explanation:

When we simply write printf("%s", p), we pass the base address of the character array to printf(). Here p+ p [3] -p[1] is passed which is p+ ('E'- 'A') . We are taking the difference of ASCII values of character E and A, but we don't need to know the exact values only knowing the difference between them will serve our purpose. Difference between 'A' and 'E' is 4 . So we are actually passing p+4 to printf. Hecne 2011 is printed.

8.

```
#include<stdio.h>

int main()
{
   char str[] = "ravindrar";
   printf("%s %s %s\n", &str[5], &5[str], str+5);
   printf("%c %c %c\n", *(str+6), str[6], 6[str]);
   return 0;
}
```

(a) Runtime Error
(b) Compiler Error
(c) rar rar rar
 r r r
(d) drar drar drar
 u u u

Solution: Option (d)

Explanation:

The program has no error. All of the following expressions mean same thing
&str[5]
&5[str]
str+5

Since compiler converts the array operation in pointers before accessing the array elements, all above result in same address.

Similarly, all of the following expressions mean same thing.
*(str+6)
str[6]
6[str]

9. In below program, what would you put in place of "?" to print "Quiz"?

```
#include <stdio.h>

int main()
{
  char arr[] = "GatesQuiz";
  printf("%s", ?);
  return 0;
}
```

(a) arr
(b) (arr+5)
(c) (arr+4)
(d) Not possible

Solution: Option (b)

Explanation:
Since %s is used, the printf statement will print everything starting from arr+5 until it finds '\0'.

10. Output?

```
int main()
{
   char a[2][3][3] = {'g','a','t','e','s','q','u','i','z'};
   printf("%s ", **a);
   return 0;
```

}

(a) Compiler Error
(b) gatesquiz followed by garbage characters
(c) gatesquiz
(d) Runtime Error

Solution: Option (c)

Explanation:
We have created a 3D array that should have 2*3*3 (= 18) elements, but we are initializing only 9 of them. In C, when we initialize less no of elements in an array all uninitialized elements become '\0′ in case of char and 0 in case of integers.

11. Consider the following C program segment:

```
char p[20];
char *s = "string";
int length = strlen(s);
int i;
for (i = 0; i < length; i++)
   p[i] = s[length — i];
printf("%s", p);
```

The output of the program is? (GATE CS 2004)

(a) gnirts
(b) gnirt
(c) string
(d) no output is printed

Solution: Option (d)

Explanation:
Let us consider below line inside the for loop
p[i] = s[length — i];
For i = 0, p[i] will be s[6 — 0] and s[6] is '\0′
So p[0] becomes '\0'. It doesn't matter what comes in p[1], p[2]…as P[0] will not change for i>0.

Nothing is printed if we print a string with first character '\0′

12.

```
#include <stdio.h>

void my_toUpper(char* str, int index)
{
   *(str + index) &= ~32;
}

int main()
{
   char* arr = "gatesquiz";
   my_toUpper(arr, 0);
   my_toUpper(arr, 5);
   printf("%s", arr);
   return 0;
}
```

(a) GatesQuiz
(b) gatesquiz
(c) Compiler dependent

Solution: Option (c)

Explanation:
The memory for the string arr is allocated in the read/write only area of data section. The choice is compiler dependent. In the newer version of compilers, the memory is allocated in the read only section of the data area. So any modification in the string is not possible.

In older version compilers like Turbo-C, modification is possible.

13. Predict the output of the following program:

```
#include <stdio.h>

int main()
{
char str[] = "%d %c", arr[] = "GatesQuiz";
printf(str, 0[arr], 2[arr + 3]);
return 0;
}
```

(a) G Q
(b) 71 81
(c) 71 Q
(d) Compile-time error

Solution: Option(c)

14. Output of following program

```
#include <stdio.h>

int fun(char *p)
{
   if (p == NULL || *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;
}
```

(a) gekskeg
(b) geeksskeeg
(c) geeks
(d) Garbage Values

Solution: Option (a)

Explanation:
The function mainly replaces more than once consecutive occurrences of a character with one occurrence.

15.

```
int main()
{
   char p[] = "gatesquiz";
   char t;
   int i, j;
   for(i=0,j=strlen(p); i<j; i++)
   {
      t = p[i];
      p[i] = p[j-i];
      p[j-i] = t;
    }
   printf("%s", p);
   return 0;
}
```

Output?

(a) ziuqsetag
(b) Nothing is printed on the screen
(c) gatesquiz
(d) gggggggg

Solution: Option (b)

Explanation:
The string termination character '\0′ is assigned to first element of array p[]

16. Assume that a character takes 1 byte. Output of following program?

```
#include<stdio.h>

int main()
{
   char str[20] = "GatesQuiz";
   printf ("%d", sizeof(str));
   return 0;
}
```

(a) 9
(b) 10

(c) 20 (d) Garbage Value

Solution: Option (c)

Explanation:
Note that the sizeof() operator would return size of array. To get size of string stored in array, we need to use strlen(). The following program prints 9.

17. Predict the output of following program, assume that a character takes 1 byte and pointer takes 4 bytes.

```
#include <stdio.h>

int main()
{
   char *str1 = "GatesQuiz";
   char str2[] = "GatesQuiz";
   printf("sizeof(str1) = %d, sizeof(str2) = %d",
         sizeof(str1), sizeof(str2));
   return 0;
}
```

(a) sizeof(str1) = 10, sizeof(str2) = 10 (b) sizeof(str1) = 4, sizeof(str2) = 10
(c) sizeof(str1) = 4, sizeof(str2) = 4 (d) sizeof(str1) = 10, sizeof(str2) = 4

Solution: Option (b)

Explanation:
str1 is a pointer and str2 is an array.

18. The output of following C program is

```
#include <stdio.h>

char str1[100];
char *fun(char str[])
{
   static int i = 0;
   if (*str)
      {
```

```
        fun(str+1);
        str1[i] = *str;
        i++;
         }
    return str1;
}

int main()
{
   char str[] = "GATE CS 2016 Mock Test";
    printf("%s", fun(str));
    return 0;
}
```

(a) GATE CS 2016 Mock Test
(b) tseT kcoM 6102 SC ETAG
(c) Nothing is printed on screen
(d) Segmentation Fault

Solution: Option (b)

Explanation:
The function basically reverses the given string.

19. Consider the following function written in the C programming language.

The output of the above function on input “ABCD EFGH” is

```
void foo (char *a)
{
  if (*a && *a != ` `)
      {
     foo(a+1);
     putchar(*a);
      }
}
```

(a) ABCD EFGH
(b) ABCD
(c) HGFE DCBA
(d) DCBA

Solution: Option (d)

20. Consider the following C program segment.

```
# include <stdio.h>

int main( )
{
   char s1[7] = "1234", *p;
   p = s1 + 2;
   *p = '\0' ;
   printf ("%s", s1);
}
```

What will be printed by the program?

(a) 12 (b) 120400
(c) 1204 (d) 1034

Solution: Option (a)

Explanation:

```
char s1[7] = "1234", *p;
   p = s1 + 2;    // p holds address of character 3
   *p = '\0' ;   // memory at s1 now becomes "12\034"
   printf ("%s", s1);  // All characters till \0 are printed
```