

Quiz 3: Set A

Introduction to Computing and Programming (CSD101)

Max. Marks: 15

Date: 21-11-2024

Duration: 30 min.

Name: _____

Roll No. _____

Q.1 struct node

```
{
```

```
    int i;
```

```
    float j;
```

```
};
```

```
struct node *s[10];
```

The above C declaration define 's' to be

(2 Marks)

- a) An array, each element of which is a pointer to a structure of type node
- b) A structure of 2 fields, each field being a pointer to an array of 10 elements
- c) A structure of 3 fields: an integer, a float, and an array of 10 elements
- d) An array, each element of which is a structure of type node.

Solution: (a)

Q.2. What does the following statement mean? **Provide justification for your answer. (2 marks)**

```
int (*fp)(char*)
```

- a. pointer to pointer
- b. pointer to an array of chars
- c. pointer to function taking a char* argument and returns an int
- d. More than one of the above
- e. None of the above

Solution: (c)

The correct answer is **pointer to function taking a char* argument and returns an int.**



Key Points

- One of the main characteristics of the function pointer is to pass the argument, and an argument can be returned from a function.
- **int (*fp)(char*)** defines that, a function pointer is declared as an integer type which is accepting a character type argument in the program.
- The main job of the function pointer is to store the initial part of the executable code.

Thus the correct answer is *pointer to function taking a char* argument and returns an int.*

Q.3. What is the use of function `char *strchr(ch, c)`? **Provide one line justification for your answer** (1 mark)

- a) return pointer to first occurrence of ch in c or NULL if not present
- b) return pointer to first occurrence of c in ch or NULL if not present
- c) return pointer to first occurrence of ch in c or ignores if not present
- d) return pointer to first occurrence of cin ch or ignores if not present

Solution: Answer: b

Explanation: The given code `char *strchr(ch, c)` return pointer to first occurrence of c in ch or NULL if not present

Q.4. What is the output of the following program? **Provide reasoning for the same. (2 marks)**

```
#include <stdio.h>

int main() {
    double a[3]={20.0,25.0,99.0}, * p,* q;
    p=a+1;
    q=p++ ;
    printf("%d,%d", (int) (q-p),( int)(* q-* p));
    return 0;}
```

Solution: Solution:

First Expression: (int)(q - p)

- q points to a[1] (value 25.0), and p now points to a[2] (value 99.0).

- The expression $q - p$ calculates the difference between the memory addresses of q and p . Since q points to $a[1]$ and p points to $a[2]$, this gives a difference of -1 (as p is after q in memory).
- Casting -1 to `int` gives -1 .

Second Expression: `(int)(*q - *p)`

- $*q$ is the value at q , which is 25.0 (the value of $a[1]$).
- $*p$ is the value at p , which is 99.0 (the value of $a[2]$).
- The expression $*q - *p$ computes $25.0 - 99.0 = -74.0$.
- Casting -74.0 to `int` gives -74 .

Q.5. Write 2 difference between `calloc()` and `malloc()`? Write the syntax for both `calloc()` and `malloc()`. **(4 marks)**

Solution:

S.No.	<code>malloc()</code>	<code>calloc()</code>
1.	A function that creates one block of memory of a fixed size.	A function that assigns a specified number of blocks of memory to a single variable.
2.	It only takes one argument	Takes two arguments.
3.	It is faster than <code>calloc</code> .	slower than <code>malloc()</code>
4.	It is used to indicate memory allocation	Used to indicate contiguous memory allocation
5.	Syntax : <code>void* malloc(size_t size);</code>	Syntax : <code>void* calloc(size_t num, size_t size);</code>
6.	It does not initialize the memory to zero	Initializes the memory to zero
7.	Does not add any extra memory overhead	Adds some extra memory overhead

Syntax of `malloc()`: `void* malloc(size_t size);`

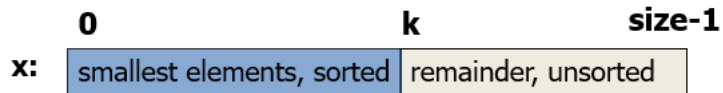
Syntax of `calloc()`: `void* calloc(size_t num, size_t size);`

Q.6 Illustrate the working of selection sort with example.

(4 marks)

Solution: If anyone has written the right code only, then also we will give full marks.

General situation :



Steps :

- Find smallest element, mval , in $x[k \dots \text{size}-1]$
- Swap smallest element with $x[k]$, then increase k .

