Programming in C++: Assignment 0

Total Marks : 00

May 13, 2020

Question 1

What is the output / Error of the following program? **MSQ**
Note: sizeof(int) = 4

```
#include <stdio.h>

main() {
    typedef int x[2];
    x myArray[3] = { 1, 2, 3, 4 }; // Line-1
    printf("%u", sizeof(myArray)); // Line-2
    printf("  %d", myArray[1][0]); // Line-3

    return 0;
}
```

a) Error in `Line-1`: wrong `typedef`, mismatch of declared size and defined size.

b) Output `Line-2`: 16

c) Output `Line-2`: 24

d) Output `Line-3`: 3

Answer: c), d)
Explanation: `typedef` is a keyword used in C language to assign alternative names to existing data types. Here `x myArray[3]` is equivalent to `int myArray[3][2]`. So, `sizeof(myArray) = 3 * 2 * 4 = 24` and `myArray[1][0]` gives output 3. For more info you may refer the following web links
https://www.geeksforgeeks.org/typedef-versus-define-c/
https://www.studytonight.com/c/typedef.php

Question 2

Which of the following is/are not a valid variable name/s in C-Language? **MSQ**

a) `int num;`

b) `float rate@12;`

c) `char* _123;`

d) `float main;`

Answer: b)
Explanation: As per the Syntax of the variable declaration in C, no special symbol, except underscore (_) is allowed in the variable name except underscore. Yes, "main" can be declared as a variable, but its value cannot be accessed.

Question 3

Find out the output / Error in the following program. **MCQ**

```
#include<stdio.h>

enum hello { a, b, c = 20.1, d };

main() {
    enum hello m = c;
    printf("%d", d);

    return 0;
}
```

a) 21

b) 3

c) 21.1

d) Compilation Error: Non-integral constant not allowed in `enum`

Answer: d)
Explanation: The above code will result a compilation Error since only integer constants are allowed in enums. For more information about `enum` you may refer the following web link *https://www.geeksforgeeks.org/enumeration-enum-c/*

Question 4

Consider the following program, identify the correct pair of truth statement if the program will be executed while it is saved with `.c` (Case-1) and `.cpp` (Case-2) extension. **MCQ**

```
#include <stdio.h>
#include <string.h>

main() {
    struct emp {
        char name[10];
        unsigned int sal;
        void TestDisp() {
            printf("%s  %u", name, sal);
        }
    };
    struct emp e1;
    strcpy(e1.name, "Employee-1");
    e1.sal = 20000;
    e1.TestDisp();

    return 0;
}
```

a) Case-1: Error: function is not allowed inside structure. Case-2: Output: `Employee-1 20000`

b) Case-1: Output: `Employee-1 20000`. Case-2: Error: function is not allowed inside structure.

c) Case-1: Error: function is not allowed in structure. Case-2: Error: function is not allowed inside structure.

d) Case-1: Output: `Employee-1 20000`. Case-2: Output: `Employee-1 20000`

Answer: a)

Explanation: Member function is not allowed inside `structure` in C-language where as it is allowed in C++. So the correct option is (a)

Question 5

What will be the output of the following Program? **MCQ**

```
#include <stdio.h>
int main() {
    int i = 0;
    for(i = 0; i < 20; i++) {
        switch (i) {
            case 0: i += 2;
            case 1: i += 5;
            case 5: i += 5;
            default: i += 3;
            break ;
        }
        printf("%d", i);
    }
    return 0;
}
```

a) 2 7 12 15 16 17 18 19 20

b) 2 5 10 13 16 19

c) 15 19

d) Compilation Error

Answer: c)
Explanation: Since there is no `break` statement in any of the `case`, all the cases starting from `0` (starting from its first true match) get executed till it gets `break` or till the end of `switch` block, without checking the case condition further. Here Loop executed twice. That is, for i = 0 (it prints 15) and for i = 16 (after the loop increment, and it prints 19).

Question 6

Consider the following linked list:

$$I \to I \to T \to K \to G \to P$$

What is the output of following function when it is called with the head of the list? **MCQ**

```
void fun(struct node* start) {
    if (start == NULL)
        return;

    printf("%c  ", start->data); // Considering data is of 'char' type

    if (start->next != NULL)
        fun(start->next->next);

    printf("%c  ", start->data);
}
```

a) I T G I G

b) I T G G

c) I T G G T I

d) I T G I T G

Answer: c)
Explanation: `fun()` prints alternate nodes of the given Linked List, first from head to end, and then from end to head. If Linked List has even number of nodes, then skips the last node.

Question 7

A single array `A[1..MAXSIZE]` is used to implement two stacks. The two stacks grow from opposite ends of the array. Variables `top1` and `top2` (`topl < top 2`) point to the location of the topmost element in each of the stacks. If the space is to be used efficiently, the condition for *stack full* is: **MCQ**

a) `(top1 = MAXSIZE/2) and (top2 = MAXSIZE/2+1)`

b) `top1 + top2 = MAXSIZE`

c) `(top1 = MAXSIZE/2) or (top2 = MAXSIZE)`

d) `top1 = top2 - 1`

Answer: d)
Explanation: If we are to use space efficiently then size of the any stack can be more than `MAXSIZE/2`. Both stacks will grow from both ends and if any of the stack top reaches near to the other top then stacks are full. So the condition will be `top1 = top2 -1` (given that `top1 < top2`)

Question 8

The seven elements A, B, C, D, E, F and G are pushed onto a stack in reverse order, i.e., starting from G. The stack is popped five times and each element is inserted into a queue. Two elements are deleted from the queue and pushed back onto the stack. Now, one element is popped from the stack. The popped item is: **MCQ**

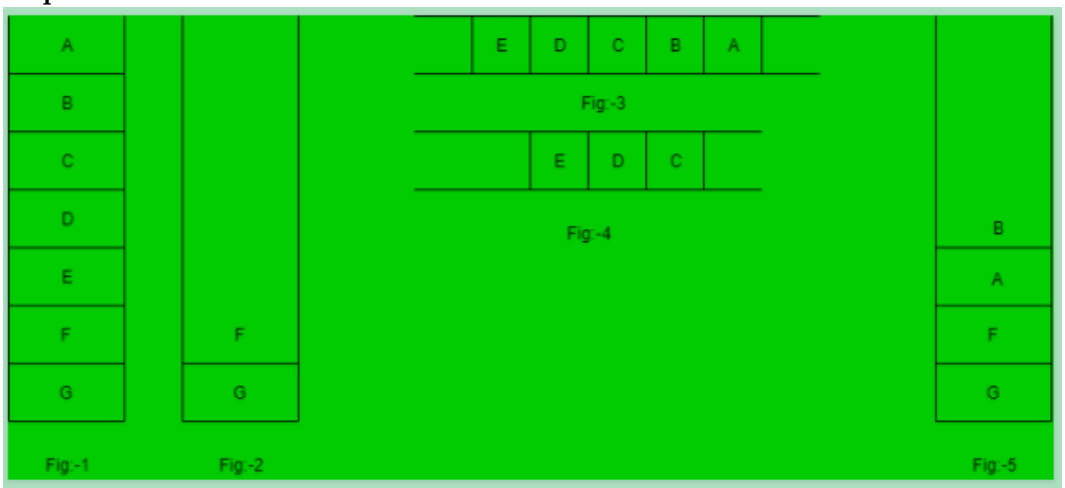
a) A

b) B

c) F

d) G

Answer: b)
Explanation:



In fig:-1 elements are inserted into a stack then in fig:-2 top 5 elements are popped and these 5 elements are inserted into a queue which is shown in fig:-3, now first two elements are deleted fro queue and pushed into stack one by one which is shown in fig:-5. At top of the stack element B is presented. So, option (b) is correct

Question 9

Consider an array `A[20][10]`, assume 4 words per memory cell and the base address of array `A` is `100`. What is the address of `A[11][5]`? Assume row major storage and index starting with zero. **MCQ**

a) 560

b) 565

c) 570

d) 575

Answer: a)
Explanation: Address (A[i][j]) = Base Address + W*(No. of columns * i + j).
Now Address (A[11][5]) = 100 + 4*(10 * 11 + 5) = 560

Question 10

Consider the following code snippet and identify the correct declaration statement/s inside function `main()`, associated with the definition of `func()`. **MSQ**

```
void func(int a) {
    printf("Value of a is %d\n", a);
}

int main() {
    void (*fun_ptr1)(int) = func;  // Statement-1
    void (*fun_ptr2)(int) = &func; // Statement-2
    void (*fun_ptr3)() = &func;    // Statement-3
    void (*fun_ptr4)() = func;     // Statement-4

    return 0;
}
```

a) Statement-1

b) Statement-2

c) Statement-3

d) Statement-4

Answer: a), b)
Explanation: Statement-1 and Statement-2 are both correct declarations of function pointer associated with `func()`. Note that a function pointer may use the name of the function (`func`) or an explicit pointer to it (`&func`).

Statement-3 and Statement-4 are wrong as these map `void` to `void` while `func` maps `int` to `void`.

Question 11

Consider an array

```
int num[ ] = {1, 2, 3, 4, 5 ,6 };
```

`p1` and `p2` are two pointers of type `(int *)`. If `p1 = num` and `p2 = p1 + 5` then what is the value of `(char*)p2 - (char*)p1`.

Note: `int` and `char` takes 4 bytes and 1 byte respectively. **MCQ**

a) 5

b) 16

c) 20

d) 24

Answer: c)
Explanation: Let us assume base address of `num = 100`, i.e `p1 = 100` and `p2 = 120`. The pointer difference value `p2-p1 = (p2-p1)/sizeof(type of data it is pointing to)`; i.e `sizeof(int) = 4` and `sizeof(char) = 1`. In our given question, during finding the difference, the pointers are type casted to 'char'. so the value of `(char*)p2 - (char*)p1`. = (120-100)/1 = 20.

Alternately, since `p1` and `p2` are of type `(int *)`, the stride is 4 bytes each. Hence 4 * 5 = 20.