Quiz 2 (September 30, 2020)

Total points 13/18

Write your name and ID correctly.
There are 18 questions in this quiz. Each question carries 1 mark. The total time duration for the quiz is 30 minutes. Answer the questions and submit your responses.

The respondent's email address (**f20181119@pilani.bits-pilani.ac.in**) was recorded on submission of this form.

0 of 0 points

Name *

Shreyas Bhat Kera

ID *

2018A7PS1119P

Questions 1-18 13 of 18 points

An array x of records is defined in a C program as follows. The character type variables are aligned in blocks of 4 contiguous bytes irrespective of their high level descriptive names while each character occupies only one byte. If the output of line 1 is 4, 4, 1 and 627601088, what is the output of line 2? 0/1

```c
#include <stdio.h>

int main()
{
    struct node{
        int A[10];
        float B[8];
        char C, D;
        char E[7];
        int F;

    } x[5];
    printf("%d %d %d %u\n", sizeof(int), sizeof(float), sizeof(char), &x);//
        Line 1
    printf("%d\n", sizeof(x)); //Line 2
    return 0;
}
```

- ○ 440
- ◉ 425
- ○ 460
- ○ None of these

Correct answer

- ◉ 440

Consider a four dimensional array A[14][16][18][7] of integers with its sizes defined as above and ranges defined in four directions R1, R2, R3 and R4. The role of R1, R2, R3, and R4 in four dimensional array resembles the notion of row and column in two dimensional arrays, where we can name 'row' as R1 and 'column' as R2. The range in each dimension starts with index 0 and ends at index one less than the size of array in the respective dimension. Consider the size of an integer as 4. If the compile time layout is implemented with R1, R2, R3, R4 form (in this order, where R4 is laid faster than R3, R3 is laid faster than R2, and so on), then the offset of element A[12][8][15][3] from base address of A is given by the number 0/1

25308

Correct answer

101232

Which of the following constructs in a C-like language is responsible for storage allocation? 1/1

- (●) Variable declaration
- ( ) None of these
- ( ) Record type definition
- ( ) Typedef construct for type renaming

Python provides comparison of two arrays A and B for testing their equality. This means 1/1

- ○ that the two array variables can have different element wise values but have their elements populated in the same storage space.
- ◉ that the elements of the two arrays are same element wise but are placed in different locations dedicated to A and B separately
- ○ that the elements of the two arrays are same element wise but are placed in same common locations accessible to both array variables
- ○ None of these

Consider the following code written in C programming language as shown. If the sizes of both integer and float are 4 bytes, and the output of line 1 is 84 and 2250686752, then what is the output of line 2 1/1

```c
#include <stdio.h>

int main()
{
    struct info{
        int x, y;
        float A[10];
        int B[7];
        float p;
        int q;
    } W[9];
    printf("%d %u\n", sizeof(W[6]),&W); //Line 1
    printf("%u\n",  &W[6].B[4]); //Line2
}
```

- ○ 2250687332
- ○ 2250686752
- ○ none of these
- ◉ 2250687320

Consider the following C code. If the sizes of integer and float are 4 bytes each, and if the output of Line 1 is 128 and 1585370112, then the output of line 2 is given below (Comma is used to add clarity only) 1/1

```c
#include <stdio.h>

int main()
{
    union info{
        int x,y;
        float z[10];
        int C[20];
    } a;
    printf("%d %u\n", sizeof(a), &a); //Line 1
    printf("%u %u %u\n", &a.z, &a.C, &a.x); //Line 2

}
```

- 1585370112, 1585370160, 1585370120
- 1585370120, 1585370160, 1585370112
- **1585370112, 1585370112, 1585370112**
- None of these

The grammar for type definition of struct construct in C language is given by the following rules. How is the derivation terminated? 1/1

<recordDef> → STRUCT ID { <fieldDeclarations> }

<fieldDeclarations> --> <oneFieldDecln> <fieldDeclarations>

<oneFieldDecln> → <type> ID ;

<type> → CHAR | INT | FLOAT

- ○ <fieldDeclarations>--> epsilon
- ○ None of these
- ◉ Both of these
- ○ <fieldDeclarations>--> <oneFieldDecln>

Consider the following C code. The character type variables are aligned in blocks of 4 contiguous bytes irrespective of their high level descriptive names while each character occupies only one byte. If the values printed as output of line 1 are 4, 4, 1 and 3106187152 respectively, the the output of line 2 is 1/1

```c
#include <stdio.h>

int main()
{
    struct node{
        int A[10];
        float B[8];
        char C, D;
        char E[7];
        int F;

    } x;
    printf("%d %d %d %u\n", sizeof(int), sizeof(float), sizeof(char), &x);//
        Line 1
    printf("%u\n",  &x.F); //Line 2
    return 0;
}
```

- 3106187240
- 3106187152
- 3106187236
- None of these

Consider the following code written in C# programming language. The array arr is 1/1

```
public class JaggedArrayTest
{
    public static void Main()
    {
        int[][] arr = new int[2][];
        arr[0] = new int[] { 11, 21, 56, 78 };
        arr[1] = new int[] { 42, 61, 37, 41, 59, 63 };
        for (int i = 0; i < arr.Length; i++)
        {
            for (int j = 0; j < arr[i].Length; j++)
            {
                System.Console.Write(arr[i][j]+" ");
            }
            System.Console.WriteLine();
        }
    }
}
```

- ○ an associative array
- ◉ a jagged array
- ○ none of these
- ○ a rectangular array

Which statements are correct? (1) In Python and Ruby programming languages, the records are implemented using hashes. (2) The record fields are always of the same size (3) One record variable in C programming language can be assigned to the other record variable of the same type. (4) One record variable in C language can be added to another variable of t he same type. (5) ADA language does not support the record data types. 0/1

- ○ Statements 2 and 5
- ○ None of these
- ◉ statements 3 and 4
- ○ statements 1 and 4
- ○ statements 2 and 4
- ○ All of these
- ○ Statements 1 and 3

Correct answer

- ◉ Statements 1 and 3

Consider the statements (1) The layout of both arrays and records is done at compile time. (2) The array element selected by A[i] cannot change at run time. (3) An array is a collection of heterogeneous elements (4) Record construct provides more flexibility in selecting the field types than arrays. Which statements are correct? 1/1

- ○ All of these
- ◉ Statements 1 and 4
- ○ Statements 2 and 3
- ○ Statements 1 and 2
- ○ Statements 1 and 3
- ○ None of these

A language supports addition A+B for array variables A and B. This means it supports 1/1

- ○ elementwise addition of all elements of the two arrays, irrespective of their sizes
- ○ concatenation of arrays by copying the elements of A into new memory locations, and then copying elements of B in the locations of A
- ○ concatenation of arrays by copying the elements of A into the memory locations of B
- ○ none of these
- ◉ elementwise addition of all elements of the two arrays, provided their sizes are same

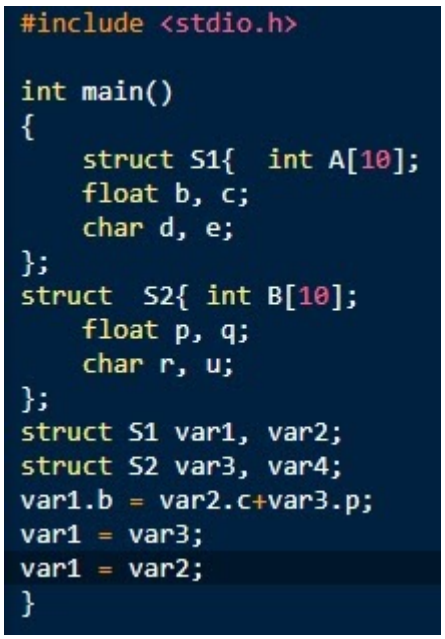
Consider the following code written in C language. Which lines will get the compile time error? 1/1

```c
#include <stdio.h>

int main()
{
    int A[10]={34, 23, 56, 46, 77, 52, 8, 90, 10, 17};
    int B[6]={33, 901, 87, 12, 78, 119};
    int *p, *q;
    int x, y;
    x=6;
    y=20;
    p=&x;
    q=&y;
    A=B; //Line 1
    p=A; //Line 2
    x=y; //Line 3
    q=&x; //Line 4
    p=q; //Line 5
    A=p; //Line 6
    x=x+1; //Line 7
    q=A+12; //Line 8
    p=p+q;//Line 9
}
```

- ○ Lines 4, 7 and 8
- ○ Lines 2, 6 and 9
- ◉ Lines 1, 6 and 9
- ○ None of these
- ○ Lines 1, 2 and 5

Consider the following C code. Which of the following is correct? 1/1



```c
#include <stdio.h>

int main()
{
    struct S1{  int A[10];
    float b, c;
    char d, e;
};
struct  S2{ int B[10];
    float p, q;
    char r, u;
};
struct S1 var1, var2;
struct S2 var3, var4;
var1.b = var2.c+var3.p;
var1 = var3;
var1 = var2;
}
```

- ○ The statement var1=var2; is type error free but the statement var1.b = var2.c+var3.p; is not
- ○ The statement var1 = var3; is type error free but the statement var1.b = var2.c+var3.p; is not
- ○ None of these
- ○ The statement var1.b = var2.c+var3.p; is type error free but the statement var1=var2; is not
- ◉ The statement var1.b = var2.c+var3.p; is type error free but the statement var1 = var3; is not,

The problem with untagged union data type is 0/1

- ○ the variable of union type is allocated memory on heap segment
- ◉ more memory requirement as compared to records
- ○ spurious data access despite the program being correct grammatically
- ○ none of these

Correct answer

- ◉ spurious data access despite the program being correct grammatically

Consider the following code in C language. If xxxx represents the spurious data, then what is the output of Line 5? (comma added for clear separation between data in each option) 1/1

```c
#include <stdio.h>

int main()
{
union point { float x; int y;} P1, P2;

P1.x = 2.6;
printf("%f \n", P1.x);  // Line 1

P1.y = 10;
printf("%d \n", P1.y);  // Line 2

P2.y = 5;
printf("%d \n", P2.y);  // Line 3

P2.x = 4.6;
printf("%f \n", P2.x);  // Line 4
printf("%f %d %d %f \n", P1.x, P1.y, P2.y, P2.x);  // Line 5
}
```

- 2.600000, xxxx, xxxx, 4.600000
- 2.600000, 10, 5, 4.600000
- xxxx, 10, 5 , xxxx
- **xxxx, 10, xxxx, 4.600000**
- none of these

The logic behind design of union data type after records type is 1/1

- ○ The user is free from the responsibility of accessing the data from allocated memory to union type variable.
- ○ none of these
- ○ that the type checking can be done at compile time, hence less execution time is needed for user code.
- ◉ that the same memory location can be used efficiently for different field data items

Consider a four dimensional array A[14][16][18][7] of integers with its sizes defined as above and ranges defined in four directions R1, R2, R3 and R4. The role of R1, R2, R3, and R4 in four dimensional array resembles the notion of row and column in two dimensional arrays, where we can name 'row' as R1 and 'column' as R2. The range in each dimension starts with index 0 and ends at index one less than the size of array in the respective dimension. Consider the size of an integer as 4. If the compile time layout is implemented with R4, R2, R1, R3 form (in this order, where R4 is laid faster than R3, R3 is laid faster than R2, and so on), then the offset of element A[12][8][15][3] from base address of A is given by the number 0/1

101232

Correct answer

57372