Midterm 1.2

(Answers)

1. What is the problem with the following code?

int i  = 5

char ptr = &i;

char \*ptr1 = &i;

printf(“%d\n”, i);

Ans: (1.5 points for identifying each error)

line 1 does not have semicolon.

line 3, we are dereferencing the pointer, which is counterproductive.

2. What are the sizes of arr and p in the C program in the 32-bit machine?

int arr[50];

int \*p = arr;

Ans: (1.5 points for each correct answer)

arr: 200 bytes

p:4 bytes

3. How many levels of pointer can we have?

Ans:

As much as a human can handle. Any c/c++ compiler definitely support more than that.

4. What is the potential address space of a system which uses 32-bit word size?

Ans:

2^32 or 4GB

5. What is the size of the following union?

1. **union** abc{
2. **int** a;
3. **char** b;
4. **float** c;
5. **double** d;
6. };
7. **int** main()
8. {
9. printf("Size of union abc is %d", **sizeof**(**union** abc));
10. **return** 0;
11. }

Ans: full score for correct answer 8 bytes which is for double

Note for TA: The size of the union is based on the size of the largest member of the union.

the size of double is 8 bytes. Since the double variable occupies the largest memory among all the four variables, so total 8 bytes will be allocated in the memory. Therefore, the output of the above program would be 8 bytes.

6. What is piping in the bash shell? Write a shell command using pipe(s).

Ans:

(3 pts)

Piping connects two commands in shell. The Unix systems allow stdout of a command to be connected to stdin of another command. You can make it do so by using the pipe character ‘|’

(2 pts)

Example: any correct command with piping

7. Why does the CPU need to be in privileged mode while OS is doing any task? Explain.

Ans:

It executes privileged instructions that interacts directly with hardware device like disks. While processes run in privileged mode or kernel mode, they have unrestricted access to hardware. In order to keep the system secured and safe, CPU allows only the processes from kernel space to access resources.

Give 3 pts if students understand privileged mode and give another 2 pts if they understand why CPU needs to be in privileged mode while OS is doing any task.

8. Tell us the difference between the following statement

|  |
| --- |
| int\* arr1[8];  int (\*arr2)[8];  int \*(arr3[8]); |

Ans: (First part: 1.5 points, second part: 1.5 points and third part: 2 points)

By C precedence table, array [], function return () have higher precedence over pointer \*.

For int\* arr1[8] arr1 first and foremost is an array no matter what type the element is. After applying pointer \*, we know arr1 is an array of int pointers.

For int (\*arr2)[8] By bracket overriding rule, pointer \* has higher precedence over array [] in this case. Then arr2 is first and foremost a pointer no matter what it is pointing to. After applying array [], we know arr2 is a pointer to an array of int.

For int \*(arr3[8]) Bracket in this case does not change any default precedence, so it is the same as int\* arr1[8]

9. Based on the code snippet, finish the function prototype for the *foo* function, clearly showing the type of each argument.

void foo(                                                                                               );

….

int main() {

  int \*a, c[10]={[0,2]=10};

  char  b;

  a = c;

  foo((float \*)&a[0], b, c);

}

Ans:

void foo( float \*a, char b, int \*c)

10. Complete line number 2, 10, 13, and 16

1. #include <stdio.h>

2. long addTwoNumbers(\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_); // Complete line 2

3. int main()

4. {

5. long fno, sno, sum;

6. printf(" Input the first number : ");

7. scanf("%ld", &fno);

8. printf(" Input the second number : ");

9. scanf("%ld", &sno);

10. sum = addTwoNumbers(\_\_\_\_\_\_); // complete line 10

11. return 0;

12. }

13. long addTwoNumbers(\_\_\_\_\_\_\_) // complete line 13

14. {

15. long sum;

16. sum = \_\_\_\_\_\_\_\_\_ // complete line 16

17. return sum;

18. }

Ans: (2 points for each line)

#include <stdio.h>

long addTwoNumbers(long \*, long \*);

int main()

{

long fno, sno, sum;

printf("\n\n Pointer : Add two numbers using call by reference:\n");

printf("-------------------------------------------------------\n");

printf(" Input the first number : ");

scanf("%ld", &fno);

printf(" Input the second number : ");

scanf("%ld", &sno);

sum = addTwoNumbers(&fno, &sno);

printf(" The sum of %ld and %ld is %ld\n\n", fno, sno, sum);

return 0;

}

long addTwoNumbers(long \*n1, long \*n2)

{

long sum;

sum = \*n1 + \*n2;

return sum;

}

11.

Write code which creates a struct type called StudentData to represent the registration data of each student in our class. The structure has two fields: the id field *stu\_id*that is an 32-bit unsigned integer, and the name field *stu\_name* that uses (up to) 20 characters.

struct StudentData {

char stu\_name[32];

unsigned int stu\_id;

} ;

Note:

· unsigned int can be uint32\_t;

· Correct struct definition, including the keyword “struct”, the name (StudentData), brackets ({}) and the semicolon (2pts)

· Correct definition of each field (1pt each)

In the above question, let us assume the struct type in addition has the third field variable *stu\_sec* to represent the student section id, which is of an enum type with three section names SECTION1 (representing 1), SECTION2 (representing 2) and SECTION3(representing 3) as its member fields. Write the code to correctly declare the enum type variable *stu\_sec*.

Ans:

enum {

SECTION1 =1,

SECTION2 =2,

SECTION3 =3

} stu\_sec;

Note:

· Correct enum definition, including the keyword “enum” and brackets ({}. (0.5 pt)

· Correctly create the variable stu\_sec (0.5 pt), NOT to use stu\_sec as the enum type num (enum stu\_sec)

· List SECTION1, SECTION2, and SECTION3 as the member field.

· Correct initialization of the fields. SECTION1 must equal 1 (if SECTION1 is correctly initialized, the other two sections will automatically be initialized as 2 and 3, respectively, so it is okay to not explicitly initialize the latter two. Finally, the different fields are separated by comma, not semi-colon;) (1pt)

Continue with the above question. Write code to (1) declare a variable *bob* and a pointer *p* of this struct type; (2) initialize *p* with the address of *bob;*(3) initialize the section id of *bob*withSECTION3 through the pointer *p*

(1) struct StudentData alice, \*p; (1 pts)

(2) p = &alice; (1 pt)

(3) p->stu\_sec= SECTION2; (2 pts)

(if students use “.” Instead of “->”, take 1 pt off. If it is not stu\_sec = SECTION2, take 2 pts off) (it is okay to assign stu\_sec =2;)