SET-A **RollNo:**

The LNM Institute of Information Technology

Quiz-2 –2016 Computer Programming(CP)

Time:30 Minute **Course Code : CSE104** **Max Marks: 15**

Q1. Find the output of the following program segments (Show the workout in the Box given) [1.5X10]

(a)

```
#include <stdio.h>
int main()
{
   char   *str="IncludeHelp";
   printf("%c\n", *str);
   return 0;
}
```

Step- I: str is a character pointer and it is pointing to 'I' of the string "IncludeHelp" .

So if I write printf("%c\n", *str);

Due to %c in printf() it will print 'I'

Output : I

(b)

```
#include <stdio.h>
      int main()
      {
         int ary[4] = {1, 2, 3, 4};
         int *p = ary + 3;
         printf("%d\n", p[-2]);
      }
```

Step 1:

3000	3004	3008	3012
1	2	3	4

Step2 : int *p = ary + 3; means

P = 3000 +3X4 (size of int)

= 3012

So pointer P is pointing to 4

Step 3: printf("%d\n", p[-2]);

Here p[-2] means *(p-2X4) where 4is sizeof int

So *(3012- 8) = *(3004)

So output will be value at address 3004

Output : 2

(c)

```
#include <stdio.h>
      void main()
      {
         char *s= "hello";
         char *p = s;
         printf("%c\t%c", 1[p], s[1]);
```

Step 1: character pointer *s is pointing to "hello"

Step 2: pointer *p is also pointing to "hello";

```
}
```

5500	5501	5502	5503	5504	5505
h	e	l	l	o	\0

Step 3: printf("%c\t%c", 1[p], s[1]);

1[p] = p[1] or = *(p+1)

Here p = 5500 (address of 'h')

So *(p+1) = *(5500+1) = *(5501)

Value at address 5501 = **'e'**

Another s[1] = *(s+1) = *(5500 +1) = *(5501)

So value at address 5501 = **'e'**

Hence: **Output : e e**

(d)

```
#include <stdio.h>
 void main()
   {
     char *s= "hello";
     char *p = s;
     printf("%c\t%c", *(p + 3),  s[1]);
    }
```

Step 1: character pointer *s is pointing to "hello"

Step 2: pointer *p is also pointing to "hello";

5500	5501	5502	5503	5504	5505
h	e	l	l	o	\0

Step 3: printf("%c\t%c", *(p + 3), s[1]);

Here p = 5500 (address of 'h')

So *(p+3) = *(5500+3) = *(5503)

Value at address 5503 = **'l'**

Another s[1] = *(s+1) = *(5500 +1) = *(5501)

	So value at address 5501 = **'e'** Hence: **Output : l e**
(e) #include<stdio.h> int main() { char *p; p="hello"; printf("%c\n", **&*&p); return 0; }	Step 1: character pointer p is pointing to string "hello" 3000 | 3001 | 3002 | 3003 | 3004 | 3005 h | e | l | l | o | \0 P 5555 Step 2: **&*&p in this statement & p = 5555 means address of pointer *&p = 3000 and then &*&p is again = 5555 *(&*&p) = 3000 and *(3000) = h **And output is : h**
(f) int check (int, int); void main() { int c; c = check(10, 20); printf("c=%d\n", c); } int check(int i, int j) { int *p, *q; p=&i; q=&j; return i>=45 ? *p : *q; }	Step 1: c = check(10, 20); function check() is called so I =10 and j = 20 int *p, *q; p=&i; p is pointing to i q=&j; q is pointing to j value of I = 10 so condition is false and hence *q will be returned from function *q means value at address q that is 20 so 20 will be returned back to main() c =20 **And output is : c =20**

(g)

```
void main()
{  int a[5] = {50, 100, 150, 200, 250};
   int i, j, m;
   i = ++a[2];
   j = a[2]+ a[1]++;
   m = a[0]+a[1];
   printf("%d, %d, %d", i, j, m);
 }
```

0	1	2	3	4
50	100	150	200	250

i = ++a[2] here we have used two operators ++ and []

[] operator has higher precedence then ++ so first of all

a[2] = 150 will be accessed then ++ will be applied so it becomes

a[2] = 151 so i = 151

j = a[2]+ a[1]++; a[2] +a[1] = 151 +100 =251

so j =251 after that a[1]++ so a[1] = 101

m = a[0]+a[1]; = 50 +101 = 151

so m = 151

so after three statement execution the array a[] become

0	1	2	3	4
50	101	151	200	250

output is : 151 251 151

(h)

```
void fun(int **ptr)
{
   **ptr=100;
}
void main()
{  int num=50;
   int *pp=&num;
   fun(&pp);
   printf("%d,%d",num,*pp);
```

num

3000

pp

7777

7777

}	ptr So in the memory scenario you can see ptr is a pointer to a pointer while pp is a integer pointer In fun(&pp) we are passing the address of pp So ptr = 7777 **ptr = 100 will change the value of num to 100 Because *ptr = 3000 means value at address 7777 **ptr means value at address *(3000) =50 and change that value to 100 **output is : 100 100**
(i) void main() { int fun(int); int i = fun(10); printf("%d\n", --i); } int fun(int i) { return (i++); }	**output is : 9**
(j) #include<stdio.h> int main() { int arr[] = {12, 13, 14, 15, 16}; printf("%d, %d, %d\n", sizeof(arr), sizeof(*arr), sizeof(arr[0])); return 0; }	Sizeof(arr) = 5X4 = 20 bytes Sizeof(*arr) = 4 because *arr is size of one integer Sizeof(arr[0]) = 4 In fact the statement *arr = arr[0] in both the cases we will get element at zeroth position **output is : 20 4 4**