C Pointer Basics

Question 11 CORRECT

```
#include<stdio.h>
void f(int *p, int *q)
{
  p = q;
  *p = 2;
}
int i = 0, j = 1;
int main()
{
  f(&i, &j);
  printf("%d %d n", i, j);
  getchar();
  return 0;
}
```

Run on IDE

A 2 2

B 2 1

C 0 1

0 2

C Pointer Basics
Discuss it

Question 11 Explanation:

See below f() with comments for explanation.

```
/* p points to i and q points to j */
void f(int *p, int *q)
{
  p = q;    /* p also points to j now */
 *p = 2;   /* Value of j is changed to 2 now */
}
```

Question 12 CORRECT

Consider this C code to swap two integers and these five statements after it:

```
void swap(int *px, int *py)
{
   *px = *px - *py;
   *py = *px + *py;
   *px = *py - *px;
}
```

Run on IDE

S1: will generate a compilation error S2: may generate a segmentation fault at runtime depending on the arguments passed S3: correctly implements the swap procedure for all input pointers referring to integers stored in memory locations accessible to the process S4: implements the swap procedure correctly for some but not all valid input pointers S5: may add or subtract integers and pointers.

A S1

B S2 and S3

S2 and S4

D S2 and S5

C Pointer Basics
Discuss it

Question 12 Explanation:
S2: May generate segmentation fault if value at pointers px or py is constant or px or py points to a memory location that is invalid S4: May not work for all inputs as arithmetic overflow can occur

Question 13 CORRECT

```
int f(int x, int *py, int **ppz)
{
  int y, z;
  **ppz += 1;
   z  = **ppz;
  *py += 2;
   y = *py;
   x += 3;
   return x + y + z;
}

void main()
{
   int c, *b, **a;
   c = 4;
   b = &c;
   a = &b;
   printf("%d ", f(c, b, a));
   return 0;
}
```

Run on IDE

A 18

19

C 21

D 22

C Pointer Basics
Discuss it

Question 13 Explanation:

Let us understand this line by line

```
/* below line changes value of c to 5. Note that x remains unaffected by
  this change as x is a copy of c and address of x is different from c*/
**ppz += 1

/* z is changed to 5*/
z  = **ppz;

/* changes c to 7, x is not changed */
*py += 2;

 /* y is changed to 7*/
y = *py;


/* x is incremented by 3 */
```

```
    x += 3;

  /* return 7 + 7 + 5*/
  return x + y + z;
```

Question 14 CORRECT

Predict the output of following program

```
#include<stdio.h>
int main()
{
    int a = 12;
    void *ptr = (int *)&a;
    printf("%d", *ptr);
    getchar();
    return 0;
}
```

Run on IDE

A 12

Compiler Error

C Runt Time Error

D 0

C Pointer Basics
Discuss it

Question 14 Explanation:
There is compiler error in line "printf("%d", *ptr);". void * type pointers cannot be de-referenced. We must type cast them before de-referencing. The following program works fine and prints 12.

```
#include<stdio.h>

int main()
{
    int a = 12;
    void *ptr = (int *)&a;
    printf("%d", *(int *)ptr);
    getchar();
```

```
    return 0;
}
```

Question 15 WRONG

```
#include<stdio.h>

void swap (char *x, char *y)
{
    char *t = x;
    x = y;
    y = t;
}

int main()
{
    char *x = "geeksquiz";
    char *y = "geeksforgeeks";
    char *t;
    swap(x, y);
    printf("(%s, %s)", x, y);
    t = x;
    x = y;
    y = t;
    printf("n(%s, %s)", x, y);
    return 0;
}
```

Run on IDE

```
(geeksquiz, geeksforgeeks)
(geeksforgeeks, geeksquiz)
```

```
(geeksforgeeks, geeksquiz)
(geeksquiz, geeksforgeeks)
```

C

```
(geeksquiz, geeksforgeeks)
(geeksquiz, geeksforgeeks)
```

D

```
(geeksforgeeks, geeksquiz)
(geeksforgeeks, geeksquiz)
```

C Pointer Basics
Discuss it

Question 15 Explanation:

See C function to Swap strings

Question 16 CORRECT

```
#include <stdio.h>
int main()
{
    int arr[] = {1, 2, 3, 4, 5};
    int *p = arr;
    ++*p;
    p += 2;
    printf("%d", *p);
    return 0;
}
```

Run on IDE

A 2

3

C 4

D Compiler Error

C Pointer Basics
Discuss it

Question 16 Explanation:
The expression ++*p is evaluated as "++(*p)" . So it increments the value of first element of array (doesn't change the pointer p). When p += 2 is done, p is changed to point to third element of array.

Question 17 CORRECT

What does the following program print?

```
#include
void f(int *p, int *q)
{
  p = q;
 *p = 2;
}
int i = 0, j = 1;
int main()
```

```
{
  f(&i, &j);
  printf("%d %d n", i, j);
  getchar();
  return 0;
}
```

Run on IDE

A 2 2

B 2 1

C 0 1

0 2

C Pointer Basics GATE CS 2010

Discuss it

Question 17 Explanation:

See below comments for explanation.

```
/* p points to i and q points to j */
void f(int *p, int *q)
{
  p = q;    /* p also points to j now */
 *p = 2;   /* Value of j is changed to 2 now */
}
```

Question 18 CORRECT

```
#include <stdio.h>
void f(char**);
int main()
{
    char *argv[] = { "ab", "cd", "ef", "gh", "ij", "kl" };
    f(argv);
    return 0;
}
void f(char **p)
{
    char *t;
    t = (p += sizeof(int))[-1];
    printf("%sn", t);
}
```

Run on IDE

A ab

B cd

C ef

gh

C Pointer Basics
Discuss it

Question 19 CORRECT

What does the following C-statement declare? [1 mark]

```
int ( * f) (int * ) ;
```

Run on IDE

A A function that takes an integer pointer as argument and returns an integer.

B A function that takes an integer as argument and returns an integer pointer.

A pointer to a function that takes an integer pointer as argument and returns an integer.

D A function that takes an integer pointer as argument and returns a function pointer

C Pointer Basics **GATE-CS-2005**
Discuss it

Question 19 Explanation:
The steps to read complicated declarations : 1)Convert C declaration to postfix format and read from left to right. 2)To convert expression to postfix, start from innermost parenthesis, If innermost parenthesis is not present then start from declarations name and go right first. When first ending parenthesis encounters then go left. Once the whole parenthesis is parsed then come out from parenthesis. 3)Continue until complete declaration has been parsed. At First, we convert the following given declaration into postfix:

```
int ( * f) (int * )
```

Since there is no innermost bracket, so first we take declaration name f, so print “f” and then go to the right, since there is nothing to parse, so go to the left. There is * at the left side, so print “*”.Come out of parenthesis. Hence postfix notation of given declaration can be written as follows:

```
f * (int * ) int
```

Meaning: f is a pointer to function (which takes one argument of int pointer type) returning int . Refer http://www.geeksforgeeks.org/complicated-declarations-in-c/ This solution is contributed by **Nirmal Bharadwaj**.

Question 20 WRONG

Consider the C program shown below.

```
#include <stdio.h>
#define print(x) printf("%d ", x)
int x;
void Q(int z)
{
    z += x;
    print(z);
}
void P(int *y)
{
    int x = *y + 2;
    Q(x);
    *y = x - 1;
    print(x);
}
main(void)
{
    x = 5;
    P(&x);
    print(x);
}
```

Run on IDE

The output of this program is

12 7 6

B 22 12 11

C 14 6 6

7 6 6

C Pointer Basics GATE-CS-2003

Discuss it

Question 20 Explanation:

x is global so first x becomes 5 by the first line in main(). Then main() calls P() with address of x.

```
// in main(void)

x = 5 // Change global x to 5
P(&x)
```

P() has a local variable named 'x' that hides global variable. P() theb calls Q() by passing value of local 'x'.

```
// In P(int *y)

int x = *y + 2; // Local x = 7
Q(x);
```

In Q(int z), z uses x which is global

```
// In Q(int z)

z += x; // z becomes 5 + 7
printz(); // prints 12
```

After end of Q(), control comes back to P(). In P(), *y (y is address of global x) is changed to x - 1 (x is local to P()).

```
// Back in P()

 *y = x - 1; // *y = 7-1
 print(x); // Prints 7
```

After end of Q(), control comes back to main(). In main(), global x is printed.

```
// Back in main()

print(x); // prints 6 (updated in P()
          //           by *y = x - 1 )
```

You have completed 20/41 questions .

Your accuracy is 85%.

1 2 3 4 5