1. Consider a situation where swap operation is very costly. Which of the following sorting algorithms should be preferred so that the number of swap operations are minimized in general?
   a) Selection sort
   b) Insertion sort
   c) Bubble sort
   d) None of the above

Solution: (a) Selection sort
Selection sort makes O(n) swaps which is minimum among all sorting algorithms mentioned above.

2. What is the advantage of bubble sort over other sorting techniques?
   a) It is faster
   b) Consumes less memory
   c) Detects whether the input is already sorted
   d) All of the mentioned

Solution: (c)
Bubble sort is one of the simplest sorting techniques and perhaps the only advantage it has over other techniques is that it can detect whether the input is already sorted.

3. How can you improve the best case efficiency in bubble sort? (The input is already sorted)

a)
```
boolean swapped = false;
for(int j=arr.length-1; j>=0 && swapped; j--)
{
        swapped = true;
        for(int k=0; k<j; k++)
        {
                if(arr[k] > arr[k+1])
                {
                        int temp = arr[k];
                        arr[k] = arr[k+1];
                        arr[k+1] = temp;
                        swapped = false;
                }
        }
}
```

b)
```
boolean swapped = true;
for(int j=arr.length-1; j>=0 && swapped; j--)
{
        swapped = false;
        for(int k=0; k<j; k++)
        {
                if(arr[k] > arr[k+1])
                {
                        int temp = arr[k];
                        arr[k] = arr[k+1];
                        arr[k+1] = temp;
                }
        }
}
```

c)
```
boolean swapped = true;
for(int j=arr.length-1; j>=0 && swapped; j--)
{
        swapped = false;
        for(int k=0; k<j; k++)
        {
                if(arr[k] > arr[k+1])
                {
                        int temp = arr[k];
                        arr[k] = arr[k+1];
                        arr[k+1] = temp;
                        swapped = true;
                }
        }
}
```

d)
```
boolean swapped = true;
for(int j=arr.length-1; j>=0 && swapped; j--)
{
        for(int k=0; k<j; k++)
        {
                if(arr[k] > arr[k+1])
                {
                        int temp = arr[k];
                        arr[k] = arr[k+1];
                        arr[k+1] = temp;
                        swapped = true;
                }
        }
}
```

Solution: (c)
A boolean variable ‘swapped’ determines whether any swapping has happened in a particular iteration, if no swapping has occurred, then the given array is sorted and no more iterations are required.

4. What is the best time complexity of bubble sort?
   a) $N^2$
   b) NlogN
   c) N
   d) $N(\log N)^2$

Solution: (c)

The bubble sort is at its best if the input data is sorted. i.e. If the input data is sorted in the same order as expected output. This can be achieved by using one boolean variable. The boolean variable is used to check whether the values are swapped at least once in the inner loop.

5. The Worst case occur in linear search algorithm when
   a) Item is somewhere in the middle of the array
   b) Item is not in the array at all
   c) Item is the last element in the array

d) Item is the last element in the array or is not there at all

Solution: (d) Item is the last element in the array or is not there at all

6. What is the output of following program?

```
# include <stdio.h>
void func(int x)
{
   x = 40;
}

int main()
{
  int y = 30;
  func(y);
  printf("%d", y);
  return 0;
}
```

a) 40
b) 30
c) Compilation error
d) Runtime error

Solution: (b) 30
Parameters are always passed by value in C. Therefore, in the above code, value of y is not modified using the function func().
Note that everything is passed by value in C. We only get the effect of pass by reference using pointers.

7. What would be the equivalent pointer expression for referring the array element a[i][j][k][l]?
   a) ((((a+i)+j)+k)+l)
   b) *(*(*(*(a+i)+j)+k)+l)
   c) (((a+i)+j)+k+l)
   d) ((a+i)+j+k+l)

Solution: (b)

8. Bisection method is used to find
   a) Derivative of a function at a given point
   b) Numerical integration of a function within a range
   c) Root of the function
   d) None of the above

Solution: (c) Root of the function

9. What is the output?

```
#include <stdio.h>
   int main()
   {
      char *s = "hello";
      char *p = s;
      printf("%c  %c", *(p + 1), s[1]);
    return 0;
   }
```

a) h e
b) e l
c) h h
d) e e

Solution: (d) e e
p points to the base address of 'hello' i.e. h. so *(p+1)= the next character i.e. e.
Similarly s[1] is also e. This is a simple example of pointer arithmetic on strings.

10. What will be output when you will execute following c code?

```
#include<stdio.h>
int main()
{
   short num[3][2]={2,7,10,12,15,18};
   printf("%d  %d",*(num+1)[1],**(num+2));
return 0;
}
```

a) 12 18
b) 18 18
c) 15 15
d) 12 15

Solution: (c) 15 15

*(num+1)[1]=*(*((num+1)+1))=*(*(num+2))=*(num[2])=num[2][0]=15
And **(num+2)=*(num[2]+0)=num[2][0]=15
This is example of pointer arithmetic on array.

11. The C-pre-processors are specified with _________symbol

a) #
b) $
c) &
d) *

Solution: (a) #
The C-preprocessors are specified with # symbol.

12. The #include directive

a) Tells the preprocessor to grab the text of a file and place it directly into the current file
b) Statements are not typically placed at the top of a program
c) All of the mentioned
d) None of the mentioned

Solution: (a)

The #include directive tells the preprocessor to grab the text of a file and place it directly into the current file and are statements are typically placed at the top of a program.

13. What will be the output?

```
#include<stdio.h>
#define X 3

int main()
{
#if !X
   printf("hello");
#else
   printf("world");

#endif
   return 0;
}
```

a) hello
b) world
c) compiler error
d) run time error

Solution: (b) world

14. Find the output of the C code given below

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

a) 2
b) 4
c) 7
d) 5

Solution: (d) The pointer p contains the address of the last element of the array. Assigning the value at that address to 5 changes the content of the array ary. The last element is replaced with 5. Thus, ary[3] is 5.

15. Which of the following does not initialize ptr to null (assuming variable declaration of a as int a=0;?

a) int *ptr = &a;
b) int *ptr = &a – &a;
c) int *ptr = a – a;
d) All of the mentioned

Solution: (a)