ASSIGNMENT SOLUTION WEEK 9

1. Which of the following are C pre-processor directive?
   a) #ifdef
   b) #define
   c) #endif
   d) all of the mentioned

   Solution: (d)

2. What is the correct order of insertion sort (in ascending order) of the array arr[5]={8 3 5 9 4}?
   a) {3 8 5 9 4}→ {3 5 8 9 4}→{3 4 5 8 9}
   b) {3 8 5 9 4}→ {3 5 8 9 4}→ {3 5 8 4 9}→ {3 5 4 8 9}→{3 4 5 8 9}
   c) {3 8 5 9 4}→{3 4 8 5 9}→{3 4 5 8 9}→{3 4 5 8 9}→{3 4 5 8 9}
   d) {8 3 5 4 9}→{8 3 4 5 9}→{3 4 5 8 9}

Solution: (a) In insertion sort, we start from the second number onwards and place it in appropriate position before its current position. Thus, the correct answer is (a).

3. Given an array arr = {12, 34, 47, 62, 85, 92, 95, 99,105} and key = 34; what are the mid values (corresponding array elements) generated in the first and second iterations of binary search?
   a) 85 and 12
   b) 85 and 34
   c) 62 and 34
   d) 62 and 47

Solution: (b) 85 and 34
In the first iteration: Mid=arr[(0+8)/2]=arr[4]=85
In the second iteration: Mid=arr[(0+3)/2]=arr[1]=34
So, the mid values are 85 and 34

4. Binary Search can be categorized into which of the following?
   a) Brute Force technique
   b) Divide and conquer
   c) Greedy algorithm
   d) Dynamic programming

Solution: (b) Divide and conquer
Since ‘mid’ is calculated for every iteration or recursion, we are diving the array into half and then try to solve the problem.

5. Select the code snippet which performs unordered linear search iteratively?

a)
```
int unorderedLinearSearch(int arr[], int size, int data)
{
    int index;
    for(int i = 0; i < size; i++)
    {
        if(arr[i] == data)
        {
            index = i;
            break;
        }
    }
    return index;
}
```

b)
```
int unorderedLinearSearch(int arr[], int size, int data)
{
    int index;
    for(int i = 0; i < size; i++)
    {
        if(arr[i] == data)
        {
            break;
        }
    }
    return index;
}
```

c)
```
int unorderedLinearSearch(int arr[], int size, int data)
{
    int index;
    for(int i = 0; i <= size; i++)
    {
        if(arr[i] == data)
        {
            index = i;
            continue;
        }
    }
    return index;
}
```

d) None of the above

Solution: (a)

Unordered term refers to the given array, that is, the elements need not be ordered. To search for an element in such an array, we need to loop through the elements until the desired element is found.

6. Consider the array A[]= {5,4,9,1,3} apply the insertion sort to sort the array . Consider the cost associated with each sort is 25 rupees, what is the total cost of the insertion sort for sorting the entire array?
    a) 25
    b) 50
    c) 75
    d) 100

   Solution: (c)

   When the element 1 reaches the first position of the array three comparisons are only required hence 25 * 3= 75 rupees.

   *step 1: 4 5 9 1 3.

   *step 2: 1 4 5 9 3

   *step 3: 1 3 4 5 9

7. What will be the output of the following C code?

```
#include <stdio.h>
#if A == 1
   #define B 0
#else
   #define B 1
#endif
int main()
{
   printf("%d", B);
   return 0;
}
```

   a) 0
   b) 1
   c) 01
   d) None of the above

Solution: (b)

Here, A is not initialized, so its default value is 0. The condition in the #if macro is FALSE and #define B 1 is executed. Therefore, B stores 1.

8. What will be the output?

```
#include <stdio.h>
#define a 10
int main()
{
  printf("%d ",a);
  int a=50;
  printf("%d ",a);
  return 0;
}
```

   a) 10 10
   b) 10 50
   c) 50 50
   d) Compilation error

Solution: (d)

If a is defined in macro, its value will be same throughput the program. The value assigned to that macro cannot be changed inside the program. This is the reason; it will show compilation error at the step
int a=50.

9. What is the worst case complexity of selection sort?

   a) O(nlogn)

   b) O(logn)

   c) O(n)

   d) O(n^2)

Solution: (d) O(n^2)

Selection sort creates a sub-list, LHS of the 'min' element is already sorted and RHS is yet to be sorted. Starting with the first element the 'min' element moves towards the final element

10. If the given input array is sorted or nearly sorted, which of the following algorithm gives the best performance?
    a) Insertion sort

b) Selection sort
c) Bubble sort
d) Quick sort

Solution: (a) Insertion sort

Insertion sort takes linear time when input array is sorted or almost sorted (maximum 1 or 2 elements are misplaced). All other sorting algorithms mentioned above will take more than linear time in their typical implementation.