Question 1 [25 Marks]

a) Provide a worst-case asymptotic time complexity of the following algorithms by using a suitable asymptotic notation considering the nearest function. Assume that there are no errors/ bugs in the algorithms. Show the meaningful work behind your answer. Marks will be deducted for direct answers. [5+5 marks]

Code	Time Complexity

`#include <iostream>`
`void complexFunction (int n) {`
`for (int i = 1; i <= n; ++i) {`
`for (int j = 1; j <= n; ++j) {`
`for (int k = 1; k <= n; k *= 2) {`
`std::cout << i * j << " ";`
`}}}}`
`int main() {`
`int n = 10;`
`complexFunction(n);`
`return 0; }`
	N^2 logn

`#include <iostream>`
`void complexFunction (int n) {`
`int i = 1;`
`while (i <= n) {`
`for (int j = 1; j <= n; ++j) {`
`for (int k = 1; k <= n; ++k) {`
`if (i % 2 == 0) {`
`std::cout << i * j * k << " ";`
`} else {`
`std::cout << -(i * j * k) << " ";`
`}`
`}`
`}`
`i++;`
`}`
`}`
`int main() {`
`int n = 5;`
`complexFunction(n);`
`return 0;`
`}`
	N^3

b) Please consider the following C++ code [3+2 marks]

```
#include <iostream>
using namespace std;

void magicFunction(int nums[], int n)
    int result[n];

    int j = 0;
    for (int i = 0; i < n ; i++)
        if (nums[i] >= 0 )
            result[j++] = nums[i];
    if (j == n || j == 0)
        return;

    for (int i = 0 ; i < n ; i++)
        if (nums[i] < 0)
            result[j++] = nums[i];

    for (int i = 0; i <= sizeof(result) / sizeof(result[0]) ; i++) {
        nums[i] = result[i];
    }
}
int main()
{
    int nums[] = {0, 9, -7, 2, -12, 11, -20};
    int n = sizeof(nums)/sizeof(nums[0]);
    cout << "Original array: ";
    for (int i=0; i < n; i++)
    cout << nums[i] <<" ";
     magicFunction(nums, n);

    cout<<"\nArray elements after magic function: ";
      for (int i=0; i < n; i++)
      cout << nums[i] <<" ";
        return 0;
}
```

i. What is the output of the above code? 3 marks

```
Original array: 0 9 -7 2 -12 11 -20
Array elements after magic function: 0 9 2 11 -7 -12 -20
```

ii. What is the purpose of the magic function? 2 marks.

Write a C++ program to move all negative elements of an array of integers to the end of the array. This is done without changing the order of the positive and negative elements of the array.

c) Please consider the following C++ code [3+2 marks]

```
#include<iostream>
using namespace std;
void magic_function_2(int nums[], int arr_size)
  {
   int i, n1, n2;

    /* There should be atleast two elements */
    if (arr_size < 2)
    {
        cout<< " Invalid Input ";
        return;
    }

    n1 = n2 = INT_MIN;
    for (i = 0; i < arr_size ; i ++)
    {

        if (nums[i] > n1)
        {
            n2 = n1;
            n1 = nums[i];
        }
        else if (nums[i] > n2 && nums[i] != n1)
        {
            n2 = nums[i];
        }
    }
    if (n2 == INT_MIN)
     {
        cout<< "No suitable output";
     }
    else
     {
        cout<<  "\n  magic_function_2 result is: " <<n2;
     }
}

int main()
{
    int nums[] = {7, 12, 9, 15, 19, 32, 56, 70};
    int               n               = sizeof(nums)/sizeof(nums[0]);
    cout << "Original array: ";
    for (int i=0; i < n; i++)
    cout << nums[i] <<" ";
    magic_function_2(nums, n);
    return 0;
}
```

i. What is the output of the above code? 3 marks

Original array: 7 12 9 15 19 32 56 70
magic_function_2 result is: 56

ii. What is the purpose of the magic function_2? 2 marks.

Write a C++ program to find the second largest element in an array of integers.

d) A double subscripted array declared as int arr [3] [5]. Please answer the following questions [1+4]

i. What is the total number of elements in the arr?

A double subscripted array int arr[3][5] has a total of 3 * 5 = 15 elements.

ii. What is the memory address of the arr [1][3] if the base address is 1010 [in decimal number]. Use formula to find the memory address. Zero marks will be awarded for direct answers.

To calculate the memory address of arr[1][3], we need to consider the size of each element in the array. Assuming each int element occupies 4 bytes of memory, and the base address of the array is 1010, we can calculate the memory address as follows:

The address of arr[1][3] can be calculated as:
Base address + (row index * number of columns + column index) * size of each element

So, the memory address of arr[1][3] would be:
1010 + (1 * 5 + 3) * 4 = 1010 + (8) * 4 = 1010 + 32 = 1042 [in decimal number].