Ques 1:

a)

$$T(n) = \begin{cases} T(n^{1/7}) & n>2 \\ 2 & n=2 \end{cases}$$

$$T(n) = T(\cancel{n^{1/7}})$$

$$T(\cancel{n^{1/7}}) = T(\cancel{n^{1/49}})$$

$$T(\cancel{n^{1/49}}) = T(n^{1/343})$$

$$\vdots$$

$$T(\cancel{2}) = 2$$

$$T(n) = 2$$

$$T(n) = O(1)$$

b)

Masters theorem

$$T(n) = aT\left(\frac{n}{b}\right) + n^k \log^p n$$

$a \geq 1$, $b > 1$, $k \geq 0$, p real no.

1) if $a > b^k$ then $T(n) = \Theta\left(n^{\log_b a}\right)$

2) if $a = b^k$

a) if $p > -1$ then $T(n) = \Theta\left(n^{\log_b a} \log^{p+1} n\right)$

b) if $p = -1$ then $T(n) = \Theta\left(n^{\log_b a} \log \log n\right)$

c) if $p < -1$ then $T(n) = \Theta\left(n^{\log_b a}\right)$

3) if $a < b^k$

a) if $p \geq 0$ then $T(n) = \Theta(n^k \log^p n)$

b) if $p < 0$ then $T(n) = O(n^k)$

$$T(n) = \begin{cases} 2T\left(\frac{n}{2}\right) + n\log n & n > 2 \\ 2 & n = 2 \end{cases}$$

$a = 2$ $b = 2$ $k = 1$ $p = 1$

$a = b^k$ and $p > -1$

$$T(n) = \Theta\left(n^{\log_b a} \log^{p+1} n\right)$$

$$T(n) = \Theta\left(n^{\log_2 2} \log^2 n\right)$$

$$T(n) = \Theta\left(n \log^2 n\right)$$

c)

```
void fun (int n)
{
    for (int i=0; i< n/2 ; i++)                  —— n/2
        for (int j=1; j+ n/2 <= n; j++)          —— n/2
            for (int k=1; k<=n; k= k*2)          —— log_2 n
                printf ("Hello");
}
```

$$TC = \frac{n}{2} * \frac{n}{2} * \log_2 n$$

$$= O\left(n^2 \log_2 n\right)$$

Ques 2 :

Array A or vector A

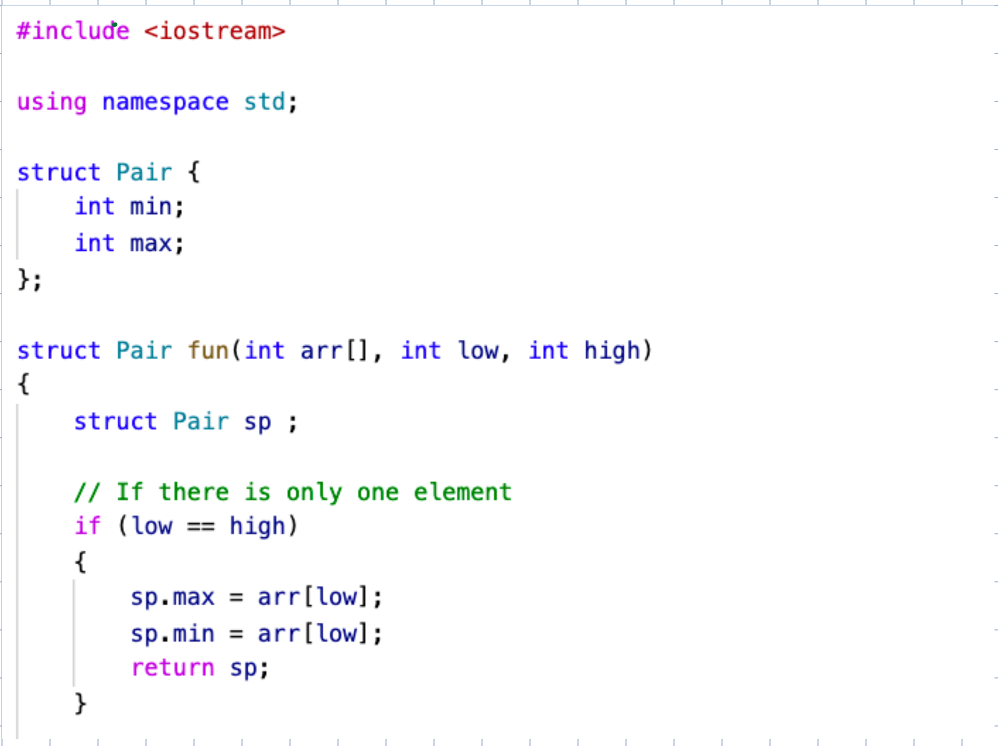
a).

```
void insert (int *A, int item)
{
        A.push_back (item)   ;
        upheapify (A.size()-1) ;
}

void upheapify (int ci)
{
        int pi= (ci-1)/2 ;

        if (data[pi] < data[ci])
        {
              swap (data[pi], data[ci]);
              upheapify (pi);
        }
}
```

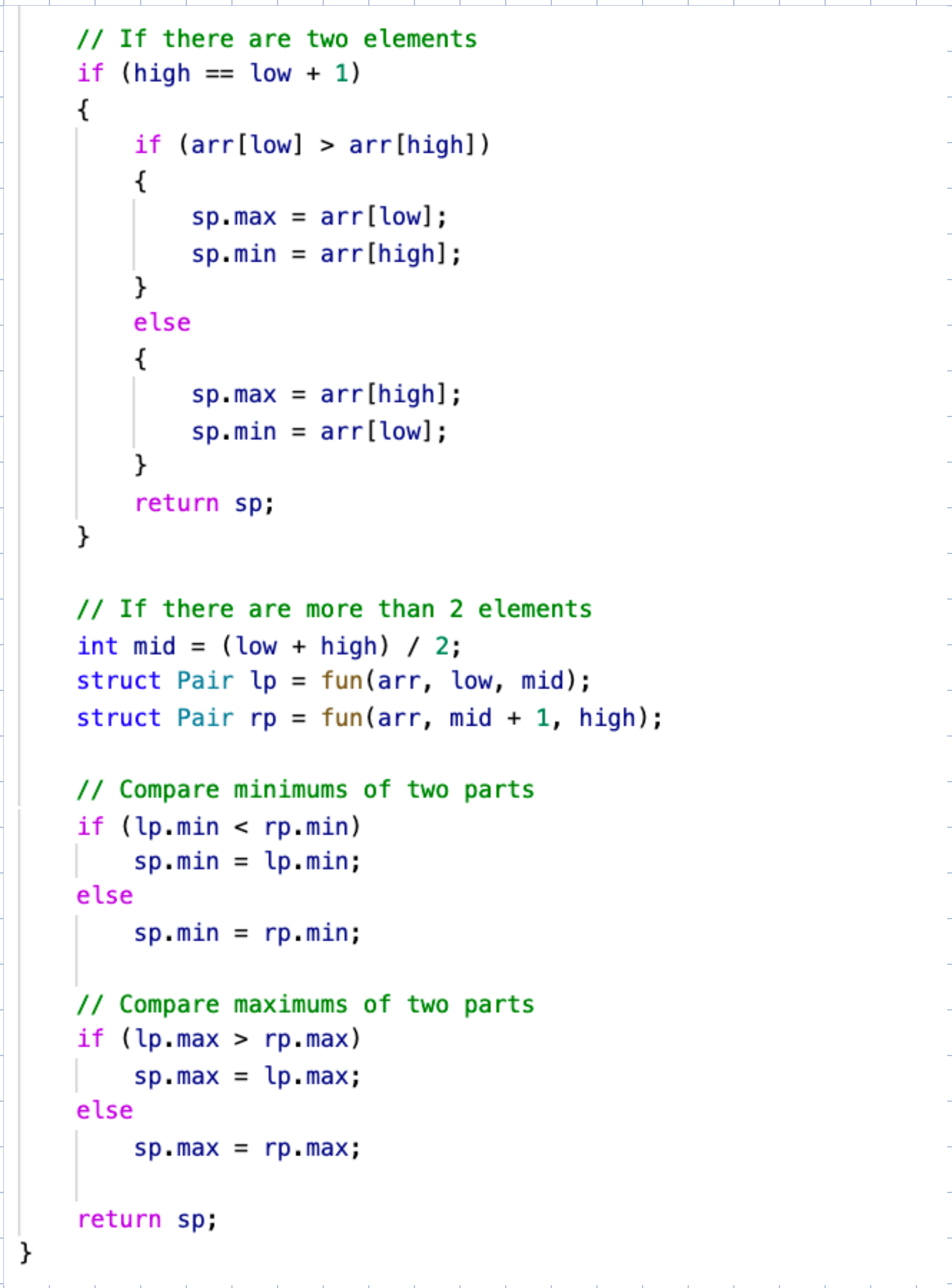
(3 Marks)

b). Divide the array in 2 parts and compare the maximum and minimum of the 2 parts to get the maximum and minimum of whole array

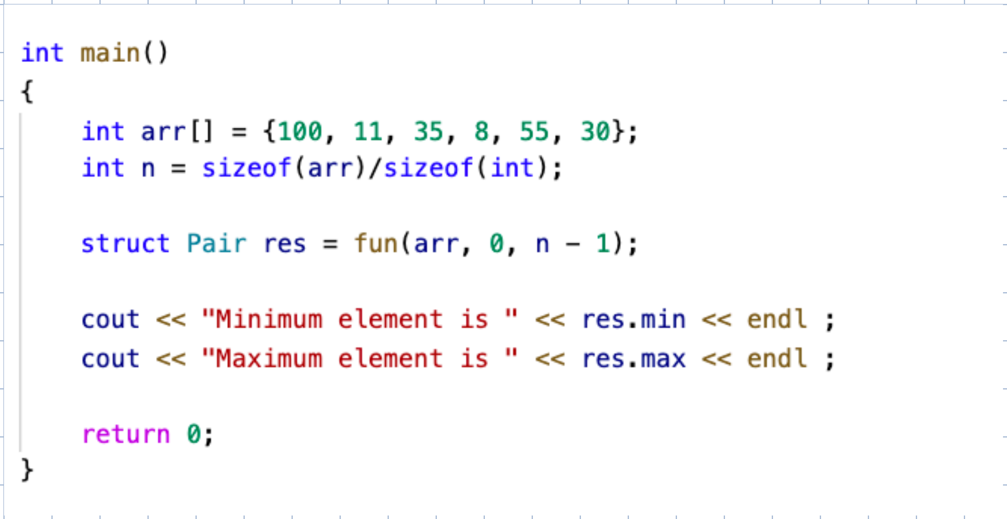
(4 Marks)



```
#include <iostream>

using namespace std;

struct Pair {
    int min;
    int max;
};

struct Pair fun(int arr[], int low, int high)
{
    struct Pair sp ;

    // If there is only one element
    if (low == high)
    {
        sp.max = arr[low];
        sp.min = arr[low];
        return sp;
    }
```



```
    // If there are two elements
    if (high == low + 1)
    {
        if (arr[low] > arr[high])
        {
            sp.max = arr[low];
            sp.min = arr[high];
        }
        else
        {
            sp.max = arr[high];
            sp.min = arr[low];
        }
        return sp;
    }

    // If there are more than 2 elements
    int mid = (low + high) / 2;
    struct Pair lp = fun(arr, low, mid);
    struct Pair rp = fun(arr, mid + 1, high);

    // Compare minimums of two parts
    if (lp.min < rp.min)
        sp.min = lp.min;
    else
        sp.min = rp.min;

    // Compare maximums of two parts
    if (lp.max > rp.max)
        sp.max = lp.max;
    else
        sp.max = rp.max;

    return sp;
}
```



```
int main()
{
    int arr[] = {100, 11, 35, 8, 55, 30};
    int n = sizeof(arr)/sizeof(int);

    struct Pair res = fun(arr, 0, n - 1);

    cout << "Minimum element is " << res.min << endl ;
    cout << "Maximum element is " << res.max << endl ;

    return 0;
}
```

Recurrence Relation : $T(n) = 2T\left(\frac{n}{2}\right) + 1$

Solving : $a=2 \quad b=2 \quad k=0 \quad p=0$

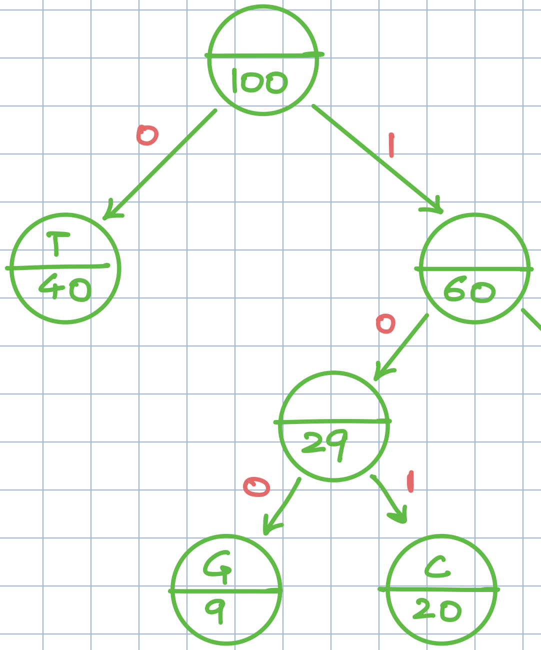
$a > b^k$
$2 > 2^0$

$T(n) = \Theta(n^{\log_b a}) = \Theta(n^{\log_2 2}) = \Theta(n)$

c)

(3 Marks)

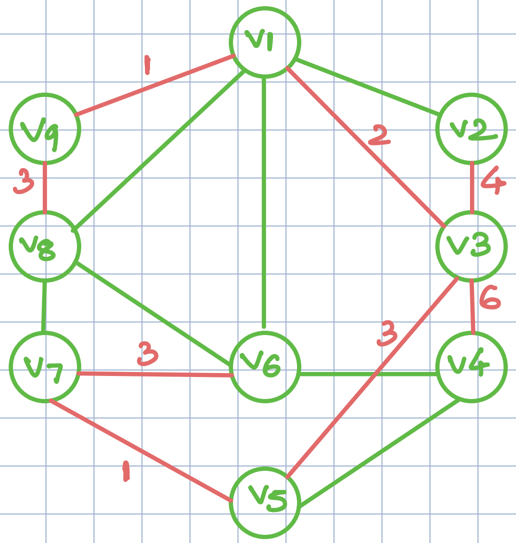
	Frequency
A	31
C	20
G	9
T	40



T : 0
G : 100
C : 101
A : 11

Ques 3 :

(7 Marks)



MST Cost = 1+1+2+3+3+3+4+6
= 23