1.

1) 문제

1. (Programming: **30 points**)
Implement the heap sort by using 'build_max_heap' function as below.
- Refer to the attach file (providing the skeleton code).
- Implement '`build_max_heap`'.
- Generate an input array of size 10, 100, 1000, and 5000, and then apply the heap sort and measure the runtime.

2) 해결방법

Build max heap 은 데이터 값의 대소관계가 아니라 array 순서대로 heap 에 추가한다. Tree 형태로 만들어진 heap 은 root 노드에 가장 큰 데이터 값이 들어가야 한다. 따라서 밑에서 2 번째 노드부터 children node 와 비교해가며 데이터 큰 값을 parent 노드로 이동시켜야 한다. Children node 의 number 는 parent number 에서 2 배씩 커지므로 leaf node 에 parent node 를 탐색할려면 heap size 에서 1/2 해주면 찾을 수 있다. 그 후 heap size*1/2 해준 값을 감소시키면서 children node 와 비교하며 교환한다. Parent node 를 Left children, right children 각각 비교 후 더 큰 데이터 값을 가진 node 를 parent node 로 올린다.

Heap sort 를 정렬해주기 위해서 delete max heap 함수를 호출해준다. Delete max heap 함수는 마지막 노드 즉 가장 큰 number 를 가진 node 를 root 노드로 올린 후 children node 중 더 큰 데이터 값을 가진 node 와 교환한다. Root 노드의 데이터 값을 반환하기 때문에 a[heap->size-1]부터 a[0]까지 채워 나가며 호출 한다.

시간 복잡도를 측정하면 0.000270second 가 나온다

3) 실행결과

```
> sh -c make -s
> ./main
Input data
8955
9439
3843
8792
660
4348
9617
9749
3751
5599

Sorted data
660
3751
3843
4348
5599
8792
8955
9439
9617
9749

Sorting result is correct.
0.000270 second
>
```

2 번

1) 문제

> 2. (Programming: **70 points**)
> Implement the code for encoding and decoding an input data using Huffman binary tree.
> - Refer to the attach file (providing the skeleton code).
> - The following functions should be implemented.
> `'huffman_traversal', 'huffman_encoding', 'huffman_decoding'`
> - It is not mandatory to follow the skeleton code. Feel free to implement your own codes, if needed.

2) 해결방법

Huffman code 는 자주 나오는 데이터 값을 가지고 코드를 만든 것이다 Huffman traversal 함수는 binary tree 를 가지고 codeword 가 무슨 데이터를 가리키고 있는 지 찾는 함수이다. 왼쪽으로 가면 0 을 오른쪽으로 가면 1 을 추가하고 leaf node 에서 데이터 값을 추출한다. If(node->l), if(node->r)을 하고 Recursive 방법으로 함수를 호출하면 leaf node 를 만날 때까지 함수가 호출된다. 이때 배열을 만들어서 왼쪽으로 가면 0 을 오른쪽으로 가면 1 을 저장한다.

Huffman encoding 은 문자열을 binary bit 으로 바꾸는 함수이다. Huffman traversal 함수에서 a,b,c,d,e,f code word 를 m_LUT[n][]에 저장하였다 문자열의 알파벳을 조사해서 m_LUT[n][]을 bits 배열에 순서대로 넣고 문자열이 끝날 때까지 반복한다.

Huffman decoding 은 bits 를 문자로 바꾸는 함수이다. Bits 배열을 가지고 0 이면 tree node 에서 왼쪽으로 가고 1 이면 오른쪽으로 가서 leaf node 의 데이터를 저장한다. 주의할 점은 데이터를 저장 후 node 를 root node 로 바꿔줘야 한다.

3) 실행결과

```
* Huffman codeword
a: 0
b: 101
c: 100
d: 111
e: 1101
f: 1100

* input chars: abacdebaf

* Huffman encoding
total length of bits stream: 23
bits stream: 01010100111110110101100

* Huffman decoding
total number of decoded chars: 9
decoded chars: abacdebaf

Process finished with exit code 0
```