Short Questions

1. (30 points)

```
#include <stdio.h>
typedef int Element;
#define LeftChild(i) (2*i+1)

/*Sort an array A of size N by constructing heap */
void HeapSort(Element A[], int N)
{
  int i;
  Element temp;
  for(i=N/2;i>=0; i--)
    PercolateDown(A,i,N);

  for (i=N-1;i>0;i--)
  {
    temp = A[i];
    A[i] = A[0];
    PercolateDown(A,0,i);
  }
  return;
}

void PercolateDown(Element A[], int i, int N)
{
  int child;
  Element temp;
  for(temp = A[i];LeftChild(i) < N; i=child)
  {
    child = LeftChild(i);
    if( (child !=N-1) && (A[child+1] > A[child]) )
    {
      child++;
    }
    if(temp < A[child])
      A[i] = A[child];
    else
      break;
  }
  A[i] = temp;
  return;
}

// test code
int main()
{
```

```
46      int A[] = {34,23,78,27,82,94,55,13,12,100};
47      int N   = 10;
48
49      HeapSort(A,N);
50      for (int i = 0; i<N;i++)
51          printf("%d ", A[i]);
52      return 0;
53  }
```

Bhondu was sleepy in class and made some mistakes while copying HeapSort code. After the class he wrote a test code to test the code he has written.

a What would be the output of his code? (20M)

b Can you please help him fix the HeapSort code he has written? (10M)

2. (30 points) H is a binary min-heap of capacity N. It is already filled in $N/2$ elements. A new random element comes in, and you need to insert it into H.

   a What is the average number of swaps required to complete this insertion in $O(\cdot)$ terms? (10M)

   b Prove the claim. (20M)

3. (40 points)

a You are given an array of elements (Size N) containing keys: true or false. Write a C code that makes this array with false keyed elements appearing before true. The algorithm should be stable and run in $O(N)$ time. (35M)

b How much extra memory you are using? (5M)