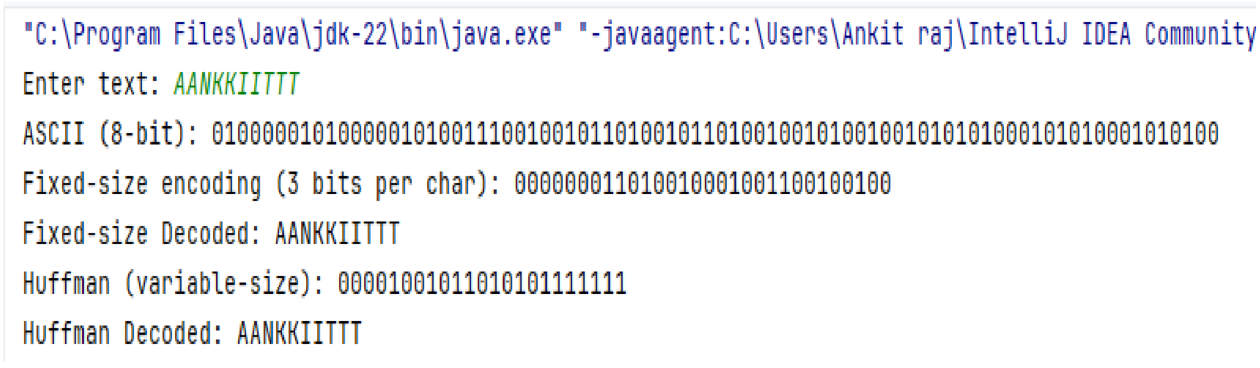
EXPERIMENT NO:-08

Aim Of Experiment:- Write a program to implement Huffman coding.

```
import java.util.*;
public class HuffmanCoding {
  static class Node implements Comparable<Node> {
    char c; int f; Node l, r;
    Node(char c, int f, Node l, Node r) { this.c = c; this.f = f; this.l = l; this.r = r; }
    public int compareTo(Node n) { return f - n.f; }
  }
  static void getCodes(Node n, String s, Map<Character, String> m) {
    if (n == null) return;
    if (n.l == null && n.r == null) { m.put(n.c, s); return; }
    getCodes(n.l, s + "0", m);
    getCodes(n.r, s + "1", m);
  }
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    System.out.print("Enter text: ");
    String txt = sc.nextLine();
    StringBuilder ascii = new StringBuilder();
    for (char c : txt.toCharArray())
      ascii.append(String.format("%8s", Integer.toBinaryString(c)).replace(' ', '0'));
    System.out.println("ASCII (8-bit): " + ascii);
    Set<Character> distinct = new TreeSet<>();
    for (char c : txt.toCharArray()) distinct.add(c);
    int nDistinct = distinct.size();
    int fixedLen = (int) Math.ceil(Math.log(nDistinct) / Math.log(2));
    if (fixedLen == 0) fixedLen = 1;
    Map<Character, String> fixedMap = new HashMap<>();
    int index = 0;
    for (Character c : distinct) {
      String code = String.format("%" + fixedLen + "s", Integer.toBinaryString(index)).replace(' ',
'0');
      fixedMap.put(c, code);
      index++;
    }
    StringBuilder fixedEncoded = new StringBuilder();
    for (char c : txt.toCharArray()) fixedEncoded.append(fixedMap.get(c));
    System.out.println("Fixed-size encoding (" + fixedLen + " bits per char): " + fixedEncoded);
    Map<String, Character> fixedInverse = new HashMap<>();
    for (Map.Entry<Character, String> e : fixedMap.entrySet())
      fixedInverse.put(e.getValue(), e.getKey());
    StringBuilder fixedDecoded = new StringBuilder();
    for (int i = 0; i < fixedEncoded.length(); i += fixedLen) {
      String chunk = fixedEncoded.substring(i, i + fixedLen);
      fixedDecoded.append(fixedInverse.get(chunk));
    }
```

```
        System.out.println("Fixed-size Decoded: " + fixedDecoded);
        Map<Character, Integer> freq = new HashMap<>();
        for (char c : txt.toCharArray())
            freq.put(c, freq.getOrDefault(c, 0) + 1);
        PriorityQueue<Node> pq = new PriorityQueue<>();
        for (var e : freq.entrySet())
            pq.add(new Node(e.getKey(), e.getValue(), null, null));
        while (pq.size() > 1) {
            Node a = pq.poll(), b = pq.poll();
            pq.add(new Node('\0', a.f + b.f, a, b));
        }
        Node root = pq.poll();
        Map<Character, String> huffCodes = new HashMap<>();
        getCodes(root, "", huffCodes);
        StringBuilder huffEncoded = new StringBuilder();
        for (char c : txt.toCharArray())
            huffEncoded.append(huffCodes.get(c));
        System.out.println("Huffman (variable-size): " + huffEncoded);
        Node cur = root;
        StringBuilder huffDecoded = new StringBuilder();
        for (char bit : huffEncoded.toString().toCharArray()) {
            cur = (bit == '0') ? cur.l : cur.r;
            if (cur.l == null && cur.r == null) {
                huffDecoded.append(cur.c);
                cur = root;
            }
        }
        System.out.println("Huffman Decoded: " + huffDecoded);
    }
}
```

OUTPUT:-



```
"C:\Program Files\Java\jdk-22\bin\java.exe" "-javaagent:C:\Users\Ankit raj\IntelliJ IDEA Community
Enter text: AANKKIITTT
ASCII (8-bit): 01000001010000010100111001001011010010110100100101001001010101000101010001010100
Fixed-size encoding (3 bits per char): 000000011010010001001100100100
Fixed-size Decoded: AANKKIITTT
Huffman (variable-size): 0000100101101010111111
Huffman Decoded: AANKKIITTT
```