9.Problem-9:

a. Description
   Find the message of the Morse code.
b. Solution-1:

c. Solution-2

```
#include<stdio.h>
#include<math.h>
#include<string.h>
#include<stdlib.h>
static const char *alpha[] = {
   ".-",   //A
   "-...", //B
   "-.-.", //C
   "-..",  //D
   ".",    //E
   "..-.", //F
   "--.",  //G
   "....", //H
   "..",   //I
   ".---", //J
   "-.-",  //K
   ".-..", //L
   "--",   //M
   "-.",   //N
```

```
    "---",  //O
    ".--.", //P
    "--.-", //Q
    ".-.",  //R
    "...",  //S
    "-",    //T
    "..-",  //U
    "...-", //V
    ".--",  //W
    "-..-", //X
    "-.--", //Y
    "--..", //Z
};
static const char *num[] = {
    "-----", //0
    ".----", //1
    "..---", //2
    "...--", //3
    "....-", //4
    ".....", //5
    "-....", //6
    "--...", //7
    "---..", //8
    "----.", //9
};
```

```
static const char **table[] = { alpha, num };


typedef enum kind {
   ALPHA, NUM
} Kind;


typedef struct mtree {
   char value;
   struct mtree *dot;
   struct mtree *bar;
} MTree;


MTree *root;


void make_tree(void);
void drop_tree(void);
void encode_out(const char *s);
void decode_out(const char *s);


int main(void){
   make_tree();
   //encode_out("HELLO WORLD");
   //encode_out("JOKE");
   decode_out("--.- ..- .- / -- --- -. / -.. ... -- / -.-. .- ---");
   //decode_out("--.- ..- .- / -- --- -. / -.. ... -- / -.-. .- ---");
```

```
    drop_tree();
    return 0;
}


void encode_out(const char *s){
    for(;;++s){
        char ch = *s;
        if(ch == '\0')
            break;
        if(isalpha(ch)){
            ch = toupper(ch);
            fputs(table[ALPHA][ch - 'A'], stdout);//`-'A'` depend on the sequence of character code
        } else if(isdigit(ch))
            fputs(table[NUM][ch - '0'], stdout);
        else if(ch == ' ')
            fputc('/', stdout);//need rest space skip ?
        else
            ;//invalid character => ignore
        fputc(' ', stdout);
    }
    fputc('\n', stdout);
}
static void decode_out_aux(MTree *tree, const char *s){
    if(tree == NULL) return;
```

```
    if(*s == '\0')
        fputc(tree->value, stdout);
    else if(*s == '/')
        fputc(' ', stdout);
    else if(*s == '.')
        decode_out_aux(tree->dot, ++s);
    else if(*s == '-')
        decode_out_aux(tree->bar, ++s);
}
void decode_out(const char *s){
    char *p;
    while(*s){
        p = strchr(s, ' ');
        if(p){
            if(p-s != 0){
                char code[p-s+1];
                memcpy(code, s, p-s);
                code[p-s]='\0';
                decode_out_aux(root, code);
            }
            s = p + 1;
        } else {
            decode_out_aux(root, s);
            break;
        }
```

```
    }
    fputc('\n', stdout);
}
static void insert_aux(MTree **tree, char ch, const char *s){
    if(*tree == NULL)
        *tree = calloc(1, sizeof(**tree));
    if(*s == '\0')
        (*tree)->value = ch;
    else if(*s == '.')
        insert_aux(&(*tree)->dot, ch, ++s);
    else if(*s == '-')
        insert_aux(&(*tree)->bar, ch, ++s);
}

static inline void insert(char ch, const char *s){
    if(*s == '.')
        insert_aux(&root->dot, ch, ++s);
    else if(*s == '-')
        insert_aux(&root->bar, ch, ++s);
}

void make_tree(void){
    root = calloc(1, sizeof(*root));
    //root->value = '/';//anything
    int i;
```

```
    for(i = 0; i < 26; ++i)
        insert('A'+i, table[ALPHA][i]);
    for(i = 0; i < 10; ++i)
        insert('0'+i, table[NUM][i]);
}
static void drop_tree_aux(MTree *root){
    if(root){
        drop_tree_aux(root->dot);
        drop_tree_aux(root->bar);
        free(root);
    }
}
void drop_tree(void){
    drop_tree_aux(root);
}
```

d. Test case