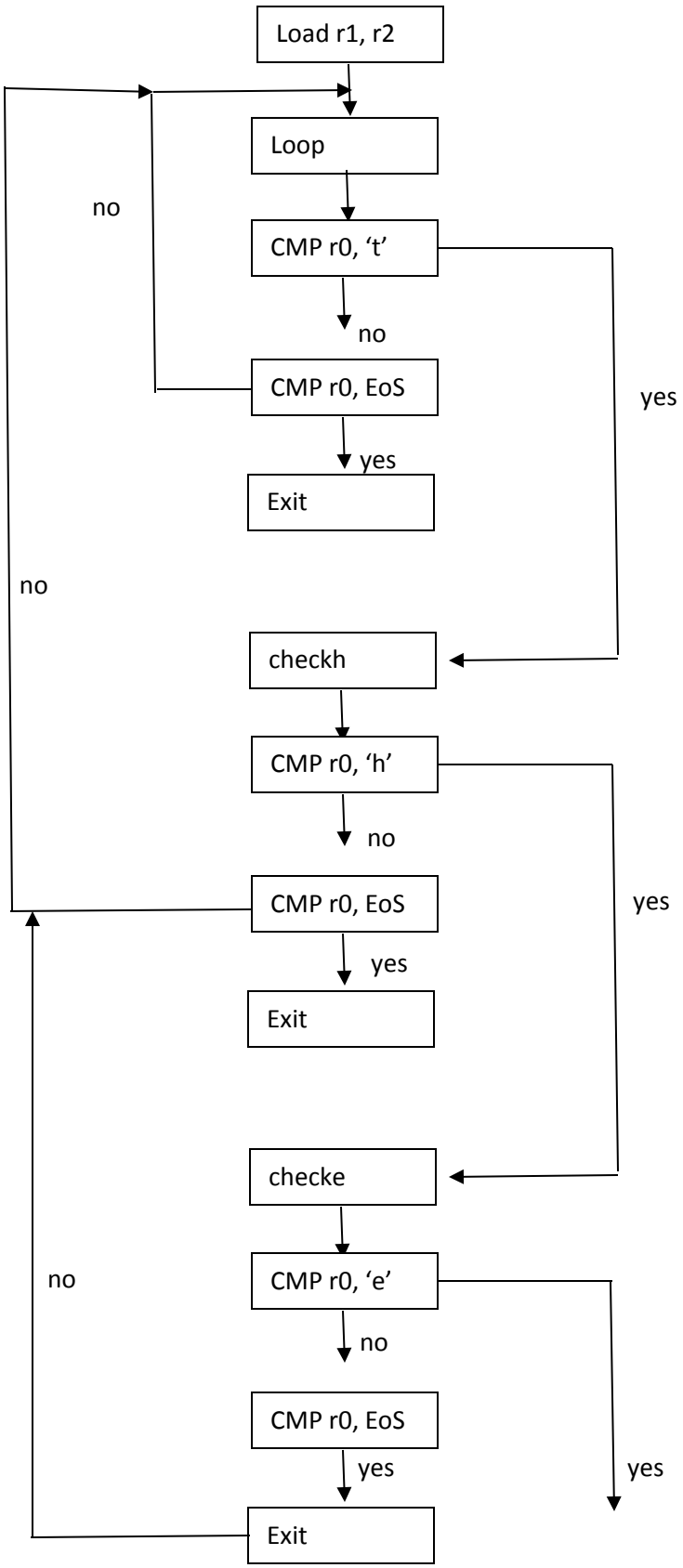
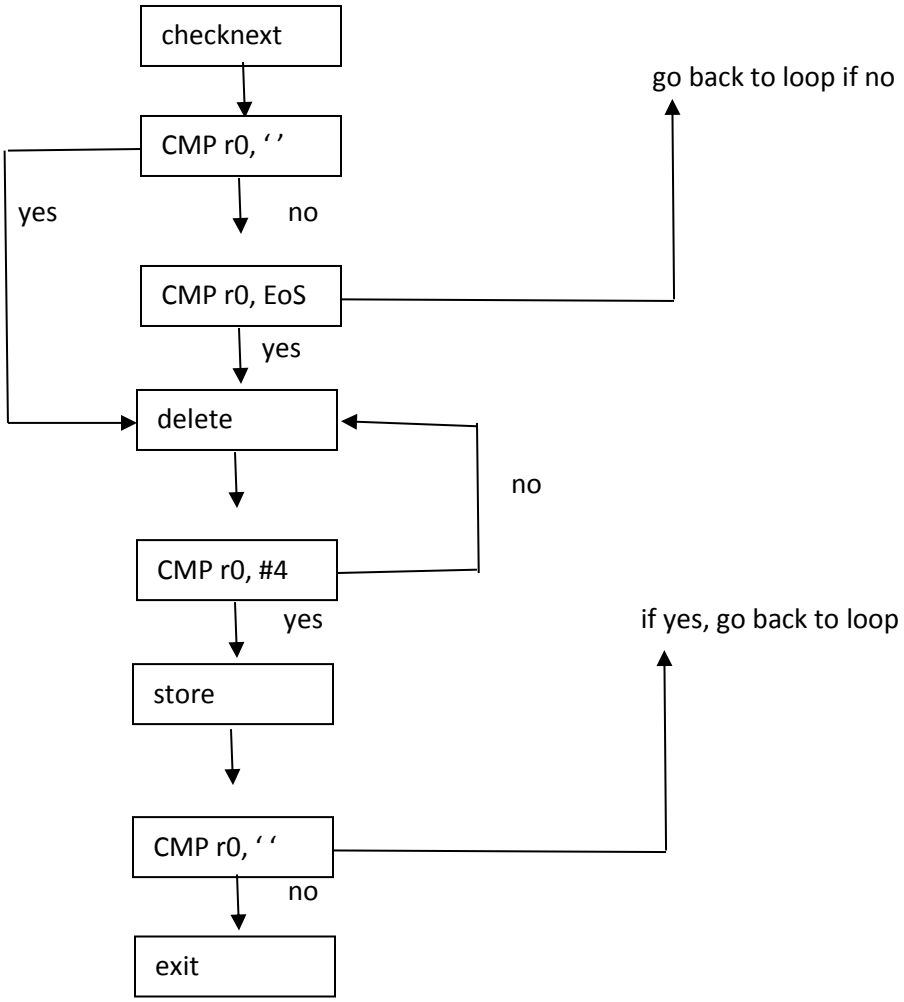

Load r1, r2
Loop
CMP r0, 't'
no
CMP r0, EoS
yes
Exit
no
yes
checkh
CMP r0, 'h'
no
CMP r0, EoS
yes
Exit
no
yes
checke
CMP r0, 'e'
no
CMP r0, EoS
yes
Exit
no
yes


checknext
CMP r0, ‘ ’
yes
no
go back to loop if no
CMP r0, EoS
yes
delete
no
CMP r0, #4
yes
if yes, go back to loop
store
CMP r0, ‘ ’
no
exit

Q1.

```
                AREA Question1, CODE, READONLY
                ENTRY

                ADR r1, STRING1         ;create a pointer point to the beginning of string1
                ADR r2, STRING2         ;create a pointer point to the beginning of string2
                ADR r0, STRING3         ;create a pointer point to the beginning of string3

LoadS1          LDRB    r3, [r1], #1    ;load the value into r3, then increase the pointer to point to the next character
                CMP     r3, #0x00       ;compare the value with EOS
                BEQ LoadS2      ;if it is EOS, then branch to string2
                STRB    r3, [r0], #1    ;if not, store the value into string3, then increase the pointer to point to the next empty position
                B       LoadS1  ;then load the next character in string1

LoadS2          LDRB r3, [r2], #1       ;load the value into r3, then increase the pointer to point to the next character
                STRB r3, [r0], #1;store the value into string3, then increase the pointer to point to the next empty position
                CMP     r3, #0x00
                BEQ Exit        ;if it is EOS, means all the characters has been stored in string3 already, so exit the program
                B       LoadS2  ;if not, then load the next character in string2

Exit            B       Exit
```

```
                    AREA Question1, DATA, READWRITE
STRING1             DCB     "This is a test string1"
EoS1        DCB     0x00
STRING2             DCB     "This is a test string2"
EoS2        DCB     0x00
STRING3             space 0xFF


                    END
```

Q2.

```
                AREA    Question2, CODE, READONLY
                ENTRY

                ADR         r1, STRING1     ;make pointer point to the first character of
string1
                ADR     r2, STRING2     ;make pointer point to the first character of string1

Loop        LDRB    r0, [r1], #1        ;load the first character of string1 to r0, then increase the
pointer
                STRB    r0, [r2], #1        ;store the character to string2
                CMP         r0, #0x74       ;check if it is 't'
                BEQ         Checkh ;if it is 't', then branch to checkh
                CMP         r0, #0x00       ;if not, check whether it is the end of string1
                BEQ         Exit    ;if it is the end of string1, then exit
                B           Loop    ;otherwise, branch to loop and read the next character

Checkh      LDRB    r0, [r1], #1        ;load the character of string1, then increase the pointer
                STRB    r0, [r2], #1        ;store the chracter to string2
                CMP         r0, #0x68       ;check if it is 'h'
                BEQ         Checke ;if it is 'h', then branch to checke
                CMP         r0, #0x00       ;if not, check whether it is the end of string1
                BEQ         Exit    ;if it is the end of string1, then exit
                B           Loop    ;otherwise, branch to loop and read the next character
```

```
Checke      LDRB    r0, [r1], #1        ;load the character of string1, then increase the pointer
            STRB    r0, [r2], #1        ;store the chracter to string2
            CMP     r0, #0x65           ;check if it is 'e'
            BEQ     CheckNext           ;if it is 'e', then branch to checknext
            CMP     r0, #0x00           ;if not, check whether it is the end of string1
            BEQ     Exit                ;if it is the end of string1, then exit
            B       Loop                ;otherwise, branch to loop and read the next character

CheckNext   LDRB    r0, [r1], #1        ;load the character of string1, then increase the pointer
            MOV     r3, #1              ;create a counter to delete the previous 3 characters in
string2, if the previous 3 characters are 'the'
            CMP     r0, #0x20           ;test if the character is blank
            BEQ     Delete              ;if it is blank, means the previous 3 characters are 'the',
then branch to delete
            CMP     r0, #0x00           ;test if the character is EoS
            BEQ     Delete              ;if it is EoS, means the previous 3 characters are 'the',
then branch to delete
            STRB    r0, [r2], #1        ;if it is neither blank or EoS, means the previous 3
characters are combined with this character, then store the character
            B       Loop                ;branch to loop to read the next character

Delete      SUB     r2, #1              ;return the pointer of string2 to the previous character
            STRB    r4, [r2]            ;set it to 0
            ADD     r3, #1              ;increase the count by 1
            CMP     r3, #4              ;test whether the delete loop has been processed 3
times
            BEQ     Store               ;branch to store, if the previous 3 characters are set to 0
```

```
                B               Delete  ;branch to store

Store           STRB    r0, [r2], #1    ;store the character to string2 after deleting
                CMP             r0, #0x20       ;test if it is blank
                BEQ             Loop    ;if it is blank, branch to loop and read the next character
in string1
                B               Exit    ;otherwise means it is EoS, so exit


Exit            B               Exit

                AREA    Question2, DATA, READWRITE
STRING1         DCB "and the man said they must go" ;String1
EoS             DCB 0x00 ;end of string1
STRING2         space 0xFF

                END
```

Q3.

```
                AREA    Question3, CODE, READONLY
                ENTRY

                MOV     r0, #3    ;create x
                ADR     sp, Stack       ;make sp points to the stack
                BL      SubR    ;call the function to do the calculation
                ADD     r1, r0, r0      ;after calculating r0, calculate r1

Exit    B       Exit

SubR    STMFD   sp!, {r1-r6, lr}    ;push elements into the stack
                LDR     r1, a     ;set a
                LDR     r2, b     ;set b
                LDR     r3, c     ;set c
                LDR     r4, d     ;set d
                MUL     r5, r0, r0      ;x*x
                MUL     r5, r1, r5      ;a*x*x
                MLA     r6, r0, r2, r3      ;b*x + c
                ADD     r0, r5, r6      ;r0 = a*x*x + b*x + c
                CMP     r0, r4    ;test whether r0 is greater than d
                MOVPL r0, r4    ;if r0 is greater, then set r0 to d
                LDMFD sp!, {r1-r6, pc}   ;return value, and reset regesters
```

```
                AREA    Question3, DATA, READWRITE
a               DCD     5
b               DCD     6
c               DCD     7
d               DCD     50
                SPACE   0xFF
Stack   DCD     0x00

                END
```