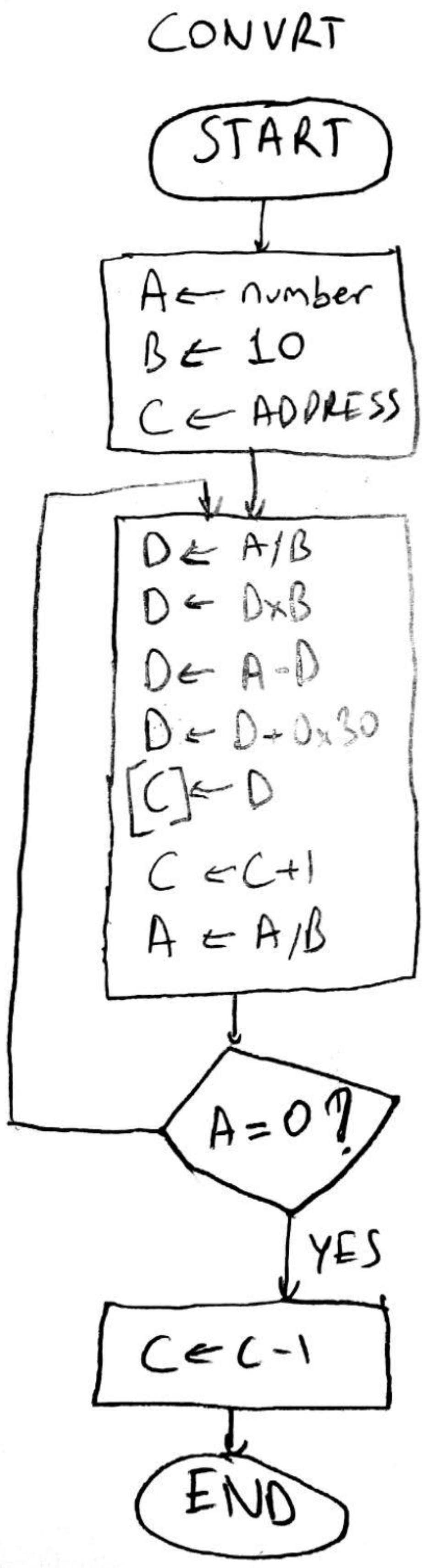
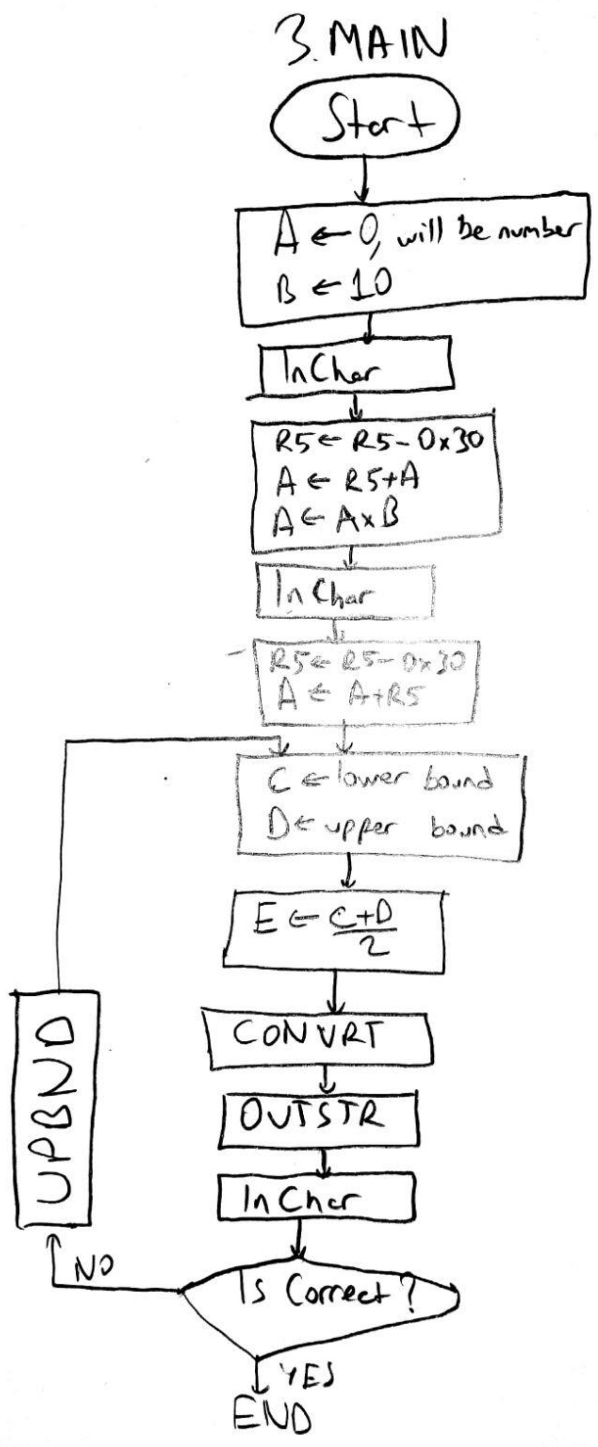
A.)FLOWCHARTS





1.CONVRT

```
;LABEL DIRECTIVE VALUE COMMENT
                AREA subroutine,READONLY,CODE
                THUMB
                EXPORT CONVRT

CONVRT PROC
                PUSH    {R0-R4}         ;keep the registers in stack
                MOV     R0,#10          ;assign 10 for decimal operations
                MOV     R2,#0x04        ;
                STRB    R2,[R5],#1      ;end of transmission
loop    UDIV    R1,R4,R0                ;divide number by 10 and keep in r1
                MUL             R1,R0   ;multiply r1 by 10
                SUB             R1,R4,R1        ;subtract r1 from r4 and keep in r1
                ADD     R1,#0x30        ;add ascii constant for numbers
                STRB    R1,[R5],#1      ;store decimal digits in [r5], increment r5
                UDIV    R4,R0           ;divide number by 10
                CMP             R4,#0   ;compare if all digits converted or not
                BEQ             last    ;if r4=0, go to last
                B               loop    ;if not repeat loop
last
                SUB             R5,#1   ;decrement r5 in order not to write 0 in front of number
                POP     {R0-R4}         ;take the registers
                BX              LR

                ENDP
                END
```

2.)MAIN

```
;LABEL DIRECTIVE VALUE COMMENT
                AREA main,CODE,READONLY
                THUMB
                EXTERN CONVRT
                EXTERN OutStr
                EXTERN InChar
                EXPORT __main
NUM             EQU             0x20004000      ;assigned address to NUM
FIRST           EQU             0x20000400      ;address for storing digits
NUMBER          EQU             0x00001275      ;desired number to convert
                ENTRY
__main
loop    BL      InChar                  ;waits any key to be pressed
                CMP     R5,#00          ;all key ascii char is different from 0
                BEQ loop                ;if different continue
                LDR R5,=FIRST           ;load address to r5
                LDR     R0,=NUM                 ;load NUM to r0
                LDR R1,=NUMBER          ;load number to r1
                STR     R1,[R0]         ;store number in the address of     NUM
                LDR     R4,[R0]         ;load number in NUM to r4
                BL      CONVRT          ;convert to decimal
                BL      OutStr          ;write to termite
                B       loop            ;infinite loop

                ALIGN
                END
```

3.)MAIN

```
;LABEL DIRECTIVE VALUE COMMENT
                AREA main,CODE,READONLY
                THUMB
                EXTERN CONVRT
                EXTERN OutStr
                EXTERN InChar
                EXTERN UPBND
                EXPORT __main
FIRST   EQU 0x20000400
__main
begin   MOV     R0,#0   ; n stored
                MOV R1,#10              ;mov 10 to r1 for decimal
                BL      InChar          ;take 2nd digit of the number
                SUB     R5,#0x30        ;delete ascii offset
                ADD     R0,R5           ;add number to r0
                MUL R0,R1               ;multiply with 10 because 2nd digit
                BL      InChar          ;take 1st digit of the number
                SUB R5,#0x30            ;delete ascii offset
                ADD     R0,R5           ;add to 2nd digit

                LDR     R2,=0x00        ;lower bound
                LDR R3,=0x01            ;upper bound
                LSL     R3,R3,R0        ;shift upper bound wrt input

recalc  ADD     R4,R3,R2                ;add upper and lower bound
                LSR     R4,R4,#1        ;divide sum with 2
                LDR     R5,=FIRST       ;load address to r5
                BL      CONVRT          ;convert number to decimal
                BL      OutStr          ;write number to port

                BL      InChar          ;UP and DOWN or C info
                CMP     R5,#0x43        ;if Correct go to begin
                BEQ     begin           ;
                BL      UPBND           ;if not correct determine new boundaries
                B       recalc          ;go to new estimation

                ALIGN
                END
```

UPBND

```
;LABEL DIRECTIVE VALUE COMMENT
                AREA subroutine,READONLY,CODE
                THUMB
                EXPORT UPBND
```

```
UPBND   PROC
                CMP             R5,#0x55        ;compare input UP
                ADDEQ   R2,R4,#1                ;if UP, add r4+1 to lower bound
                CMP             R5,#0x44        ;compare input DOWN
                SUBEQ   R3,R4,#1                ;if DOWN, subtract r4+1 to upper bound
                BX              LR             ;go to next command in main

                ALIGN
                ENDP
```

4.)MAIN

```
;LABEL DIRECTIVE VALUE COMMENT
                AREA sdata, DATA, READONLY
                THUMB

correct DCB "Palindrome"
                DCB     0x04
NOTcor  DCB "Not Palindrome"
                DCB     0x04
                AREA main,CODE,READONLY
                THUMB
                EXTERN InChar
                EXTERN OutStr
FIRST   EQU 0x20000400
                EXPORT __main

__main
L1              MOV     R0,#0
                LDR     R1,=FIRST
                LDR     R2,=FIRST
loop    BL      InChar                  ;take digit
                CMP     R5,#0x3A        ;compare with :(0x3A in ascii)
                BEQ     label           ;
                SUB     R5,R5,#0x30     ;if number continues, delete ascii offset
                STRB R5,[R1],#1   ;store digit in [r1]
                B       loop            ;take next digit
label   SUB     R1,#1                   ;decrease r1
loop2   LDRB    R3,[R1],#-1             ;load [r1] to r3,decrement r1
                LDRB    R4,[R2],#1      ;load [r2] to r4, increment r2
                CMP     R3,R4                   ;compare digits
                BNE NOT                 ;if not equal go to not correct
                CMP     R1,R2           ;if equal compare pointers
                BCS     loop2           ;if r1 is higher or same repeat
                LDR     R5,=correct     ;if not, it finishes
                B       last
NOT             LDR     R5,=NOTcor
last    BL      OutStr                  ;print determined string
                B       L1              ;infinite loop
                END
```

5.)MAIN

```
;LABEL DIRECTIVE VALUE COMMENT
            AREA main,CODE,READONLY
            THUMB
            EXTERN InChar
            EXTERN OutStr
            EXTERN CONVRT
            EXTERN PORTAL
FIRST   EQU 0x20000400
            EXPORT __main

__main
MOVE    MOV     R0,#0   ;input number
            MOV     R1,#10  ;10 for decimal

LOO         BL      InChar          ;inputchar
            CMP     R5,#0x3A        ;compare input with ':'
            BEQ     L1              ;if equal, complete
            MUL R0,R1               ;if not, multiply with current number with 10
            SUB     R5,#0x30        ;delete ascii constant
            ADD     R0,R5           ;add to number
            B       LOO             ;repeat the procedure
L1          MOV     R6,R0           ;r6 keep number with no change
            BL      PORTAL          ;Main operation

            MOV     R4,R6           ;load number to r4
            LDR     R5,=0x20000400
            BL      CONVRT;convert to decimal
            BL      OutStr          ;show the decimal number
            B       MOVE            ;go to start
            END
```

PORTAL

```
;LABEL DIRECTIVE VALUE COMMENT
            AREA subroutine,READONLY,CODE
            THUMB
            EXPORT PORTAL

PORTAL      PROC
            CMP     R0,#0
            BEQ     GOEND
            MOV     R1,#0           ;r1 will keep condition for all
            CMP     R0,#99          ;compare with 99 for p1
            BLS     PL1             ;if less skip
            ADD     R1,#8   ;COND FOR P1
PL1         ANDS R2,R0,#1           ;even-odd
            BNE     PL2             ;if not equal skip
            ADD R1,#2                ;COND FOR P3
            B       PL3
PL2         CMP     R0,#50          ;compare with 50 for p2
```

```
                BLS     PL3             ;if less skip
                ADD     R1,#4            ;COND FOR P2
PL3             MOV     R2,#7           ;dividing by 7
                UDIV    R3,R0,R2
                MUL     R3,R2
                SUBS    R3,R0
                BNE             OUT1
                ADD     R1,#1           ;COND FOR P4

OUT1    ANDS    R4,R1,#1                ;portal4
                CMP     R4,#1
                BEQ     PORT4
                ANDS    R4,R1,#8        ;portal1
                CMP     R4,#8
                BEQ     PORT1
                ANDS    R4,R1,#4        ;portal2
                CMP     R4,#4
                BEQ     PORT2
                ANDS    R4,R1,#2        ;portal3
                CMP     R4,#2
                BEQ     PORT3
                B       GOEND

PORT1   PUSH{R0}
                SUB     R0,#47
                SUB     R1,#8
                PUSH{R1}
                PUSH {LR}
                BL      PORTAL
                POP{LR}
                POP{R1}
                POP{R0}
                B       OUT1
PORT2   PUSH{R0}
                PUSH{R0}
                MOV     R5,#10
                MOV     R7,#1
PORT2LOOP               UDIV R8,R0,R5
                CMP     R0,#0
                BEQ     PORT2END
                MLS     R9,R8,R5,R0
                MOV     R0,R8
                CMP     R9,#0
                BEQ     PORT2LOOP
                MUL     R7,R9
                B       PORT2LOOP
PORT2END        POP{R0}
                SUB     R0,R7
                SUB     R1,#4
                PUSH{R1}
```

```
            PUSH{LR}
            BL      PORTAL
            POP{LR}
            POP{R1}
            POP{R0}
            B       OUT1
PORT3   PUSH{R0}
            LSR     R0,#1
            SUB     R1,#2
            PUSH{R1}
            PUSH{LR}
            BL      PORTAL
            POP{LR}
            POP{R1}
            POP{R0}
            B       OUT1
PORT4   PUSH{R0}
            MOV     R4,#3
            UDIV    R5,R0,R4
            MUL     R5,R4
            SUB     R0,R5
            SUB     R1,#1
            PUSH{R1}
            PUSH{LR}
            BL      PORTAL
            POP{LR}
            POP{R1}
            POP{R0}
            B       OUT1

GOEND       CMP     R6,R0
            BLS     L5
            MOV     R6,R0
L5          BX      LR
            ENDP
            END
```