1) Write an ALP to mov of data from memory location X to memory location Y without overlap.

```
ORG 00H
MOV R0,#30H
MOV R1,#40H
MOV R2,#06H
REP:MOV A,@R0
MOV @R1,A
INC R0
INC R1
DJNZ R2,REP
END
```

2)write an ALP to add N number of data bytes available from memory location X.store the 16 bit result in internal RAM location X and X+1.

```
ORG 00H
MOV R0,#30H
MOV A,@R0
MOV R2,A
MOV R3,#00H
BACK:INC R0
ADD A,@R0
JNC SKIP
INC R3
SKIP:DJNZ R2,BACK
MOV P0,A
MOV P1,A
END
```

3)Write an ALP to exchange N bytes of data stored from memory location X with N bytes of data stored from memory location Y

```
ORG 00H
MOV R0,#30H
MOV R1,#40H
MOV R2,#06H
BACK:MOV A,@R0
XCH A,@R1
XCH A,@R0
INC R0
INC R1
```

```
DJNZ R2,BACK

END
```

4)write an ALP to search the given number in a block of N bytes of data stored from Memory location X+1 and the length of memory location is stored in memory location X. if the search is successful display the number at port0,otherwise display zero.

```
ORG 00H

BYTE EQU 50H

MOV R0,#30H

MOV R2,#05H

BACK:MOV A,@R0

CJNE A,BYTE,SKIP

MOV P0,A

SJMP HERE

SKIP:INC R0

DJNZ R2,BACK

MOV P0,#00H

HERE:SJMP HERE

END
```

5)Write an ALP to find the largest/smallest number in an array of data available from memory location X and the length of the array is available in memory location LENGTH. Display the result in memory location LARGE/SMALL.

```
ORG 00H

RESULT EQU 60H

MOV R0,#30H

MOV R2,#08

MOV A,@R0

REP:INC R0

MOV B,@R0

CLR C

SUBB A,B

JNC SKIP

MOV A,B

SJMP FRONT

SKIP:ADD A,B

FRONT:DJNZ R2,REP

MOV RESULT,A

END
```

6)write an ALP to find the given byte of data is PALLINDROME or not. If palindrome displays FF at port1 otherwise display 00.

```
ORG 00H
PALIN EQU 50H
MOV A,PALIN
MOV R0,#00H
MOV R1,#00H
MOV R2,#08H
CLR C
REP:RLC A
MOV R1,A
MOV A,R0
RRC A
MOV R0,A
MOV A,R1
DJNZ R2,REP
MOV A,R0
CJNE A,PALIN,NPAL
MOV P2,#0FFH
HERE:SJMP HERE
NPAL:MOV P2,#00H
END
```

7) write an ALP find the number whose 7^{th} bit is given by A7=A5+A4+A6. Display the result at port 0

```
ORG 00H
MOV A,#76H
MOV R0,A
ANL A,#64H
JZ NOCHANGE
MOV A,R0
ORL A,#80H
MOV P0,A
MOV A,R0
MOV P1,A
SJMP FRONT
NOCHANGE: ORL A,#7FH
MOV P0,A
MOV A,R0
```

```
MOV P1,A
FRONT:SJMP FRONT
END
```

8)write an ALP to check whether a given byte stored at location X belongs 2 out of 5 codes. The code is valid if 3 MSB's are zeros and the number of one's in the remaining 5 numbers is two. Display FF if valid otherwise 00 in data field.

```
ORG 00H
MOV R0,#30H
MOV A,@R0
ANL A,#0E0H
JZ FRONT
NVALID:MOV P0,#00H
SJMP HERE
FRONT:MOV R1,#00H
MOV A,@R0
MOV R2,#05H
REP: RRC A
JNC SKIP
INC R1
SKIP:DJNZ R2,REP
CJNE R2,#02H,NVALID
MOV P0,#0FFH
HERE:SJMP HERE
END
```

9)write an ALP for multibyte addition the members the stored from memory location X and Y. display the result from location Z.

```
ORG 00H
MOV R0,#30H
MOV R1,#40H
MOV R2,#06H
SETB PSW.3
MOV R0,#50H
CLR PSW.3
CLR C
BACK:MOV A,@R0
ADDC A,@R1
SETB PSW.3
```

```
MOV @R0,A
INC R0
CLR PSW.3
SKIP:INC R0
INC R1
DJNZ R2,BACK
JNC SKIP
CLR A
SETB PSW.3
END
```

10)Write an ALP to arrange a block of data in ascending/ descending order.

```
ORG 00H
MOV R1,#09H
AGAIN:MOV R0,#30H
MOV R2,#09H
UP:MOV A,@R0
INC R0
MOV B,@R0
CLR C
SUBB A,B
JC SKIP
DEC R0
MOV A,@R0
MOV @R0,B
INC R0
MOV @R0,A
SKIP:DJNZ R2,UP
DJNZ R1,AGAIN
HERE:SJMP HERE
END
```

11)Write an ALP to convert the given two digit BCD number to binary and display the value at port 0.

```
ORG 00H
BCD EQU 50H
BIN EQU 51H
MOV A,BCD
MOV R0,A
```

```
ANL A,#0FH
MOV R1,A
MOV A,R0
ANL A,#0F0H
SWAP A
MOV B,#0AH
MUL AB
ADD A,R1
MOV P0,A
END
```

12)write an ALP to convert the given number into BCD number. store the result in consecutive memory location.

```
ORG 00H
BIN EQU 50H
BCD1 EQU 50H
BCD2 EQU 51H
BCD3 EQU 53H
MOV A,BIN
CJNE A,#64,DTEN
JC FRONT
DHUN:MOV B,#64H
DIV AB
MOV BCD1,A
MOV A,B
SJMP FRONT
DTEN:JNC DHUN
MOV A,BIN
FRONT:MOV B,#0AH
DIV AB
MOV BCD2,A
MOV BCD3,A
END
```

13)Write an ALP to convert 2 digit BCD number into ASCII number. Store the result in memory location ASCI1 and ASCI2.

```
ORG 00H
BCD EQU 50H
ASCI1 EQU 51H
```

```
ASCI2 EQU 52H
MOV A,BCD
ANL A,#OFH
ADD A,#30H
MOV ASCI1,A
MOV A,BCD
ANL A,#0F0H
SWAP A
ADD A,#30H
MOV ASCI2,A
END
```

C PROGRAM

```
#include<reg52.h>
#define DEL 30000
#define LOWER 0x00
#define UPPER 100
unsigned char binbcd(unsigned char );
void delay(unsigned int  del){
     while(del--);
     }
void main (void)
{
     unsigned char val;
     while(1)
     {
          for(val = LOWER;val < UPPER;val++)
          {
               P0 = binbcd(val);
               delay(DEL);
          }
     }
}
unsigned char binbcd(unsigned char i)
{
     return(((i / 10) << 4) | (i % 10));
}
```