UVA 136 Assembly

```
include "emu8086.inc"
.model small
.stack 100h
.code
main proc
printn "The 1500'th ugly number is 859963392."

mov ah,4ch
int 21h
end main
end main
```

UVA 136 C

```
#include<stdio.h>
int main()
{
    printf("The 1500'th ugly number is
859963392.\n");
    return 0;
}
```

| UVA 264 Assembly | jge next |
|--|--|
| include "emu8086.inc" | sub bl,bh ;value-i mov value,bl |
| .model small | inc bh ;i++ |
| .stack 100h | mov i,bh |
| .data | jmp for_loop |
| info db "The input list contains a single number | next: |
| per line and",10,13," will be terminated by 0.\$" | getting the mod value of i%2 store in ah |
| value db? | mov al,i |
| a db? | mov bl,2 |
| b db? | div bl |
| i db? | cmp ah,1 ;checking ah==1? |
| n db? | je work1 |
| .code | mov al,value |
| main proc | mov a,al ;a=value |
| ;fetching data segment | jmp work2 |
| mov ax,@data | work1: |
| mov ds,ax | ;a=1+i-value |
| ;printing info | mov bl,1 |
| lea dx,info | mov bh,i |
| mov ah,9 | add bl,bh |
| int 21h | sub bl,value |
| printn | mov a,bl |
| mov value,0 ;value=0 | work2: |
| again: | ; b=i-a+1; |
| input_for: | mov bh,i |
| ;user input | mov bl,a |
| mov ah,1 | sub bh,bl |
| int 21h | add bh,1 |
| cmp al,13 ;checking input==\n | mov b,bh |
| je next_loop | ;printing the result in this formation |
| taking multi digit input; | ;printf("TERM %d IS %d/%d\n",n,a,b); |
| sub al,48 | printn |
| mov bl,al | print "TERM " |
| mov al,value | mov bh,n |
| mov ah,10 | cmp bh,9 |
| mul ah | jg twodigit_value |
| add al,bl | mov ah,2 |
| mov value,al | mov dl,n |
| jmp input_for | add dl,48 |
| next_loop: | int 21h |
| mov al,value ;checking input==0? | jmp is |
| cmp al,0 | twodigit_value: |
| je exit ;if true then exit | mov al,0 |
| mov value,al | add al,n |
| mov n,al | mov ah,0 |
| mov bh,1 ;for(int i=1 | aaa |
| ;making the operation with for loop | mov bx,ax |
| for_loop: | mov ah,2 |
| mov bl,value | mov dl,bh |
| cmp bh,bl ;i <value< td=""><td>add dl,48</td></value<> | add dl,48 |

```
int 21h
  mov dl,bl
  add dl,48
  int 21h
 is:
  print " IS "
  mov ah,2
  mov dl,a
  add dl,48
  int 21h
  print "/"
  mov ah,2
  mov dl,b
  add dl,48
  int 21h
  printn
     ;making all variable value to 0
  mov value,0
  mov i,0
  mov a,0
  mov b,0
  jmp again ;calling again for restarting the
program
 exit:
  mov ah,4ch
  int 21h
  main endp
end main
```

UVA 264 C

```
#include<stdio.h>
int main()
{
    int a,b,n,i,value;
    while(scanf("%d",&n)!=EOF)
    {
       value=n;
       for(i=1; value>i; i++)
           value-=i;
       if(i%2==1)
           a=1+i-value;
       else
           a=value;
       b=i-a+1;
       printf("TERM %d IS %d/%d\n",n,a,b);
    }
    return 0;
}
```

| UVA 382 Assembly | div bl |
|---|--|
| | cmp ah,0 |
| include "emu8086.inc" | je addition ;jumping addition loop |
| .model small | if(value%2==0) |
| .stack 100h | inc bl ;i++ |
| .data | jmp loop_for |
| value db? | |
| result db? | ;result+=i; |
| length db? | addition: |
| .code | mov bh,result |
| main proc | add bh,bl |
| ;fetching data from data segment | mov result,bh |
| mov ax,@data | inc bl |
| mov ds,ax | jmp loop_for |
| printn "PERFECTION OUTPUT" | |
| again: | real_ans: |
| ;making all the variable value to zero | mov bh,result |
| mov result,0 | cmp bh,value |
| mov value,0 | je equal ;if(sum==a) jmp equal |
| taking input; | cmp bh,value ;if(sum <a)< td=""></a)<> |
| ;scanf for two digit | jl lesser ;jmp lesser |
| loop_input: | ,, ,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,, |
| mov ah,1 | |
| int 21h | ;else printf("%5d ABUNDANT\n",a); |
| cmp al,13 ;checking whether it is new line | mov bh,value |
| je next | cmp bh,9 |
| sub al,48 | jg greater1 |
| mov bl,al | mov ah,2 |
| mov al,value | mov dl,value |
| mov bh,10 | add dl,48 |
| mul bh | int 21h |
| add al,bl | printn " ABUNDANT" |
| mov value,al | jmp again |
| | Jiiip agaiii |
| jmp loop_input | for two digit output |
| novt: | ;for two digit output |
| next: | greater1: |
| printn | mov al,value |
| ;input value/2 | mov ah,0 |
| mov al,value | aaa |
| cmp al,0 | mov bx,ax |
| je exit | mov ah,2 |
| mov ah,0 | mov dl,bh |
| mov bl,2 | add dl,48 |
| div bl | int 21h |
| mov length,al | mov dl,bl |
| mov bl,1 ;for(int i=1 | add dl,48 |
| loop_for: | int 21h |
| cmp bl,length ;i <value 2<="" td=""><td>printn " ABUNDANT"</td></value> | printn " ABUNDANT" |
| jg real_ans | jmp again |
| mov al, value | |
| mov ah,0 | ;printf("%5d PERFECT\n",a); |

```
equal:
mov bh, value
cmp bh,9
jg greater2
mov ah,2
mov dl,value
add dl,48
int 21h
printn " PERFECT"
jmp again
     ;for two digit output
greater2:
mov al, value
mov ah,0
aaa
mov bx,ax
mov ah,2
mov dl,bh
add dl,48
int 21h
mov dl,bl
add dl,48
int 21h
printn " PERFECT"
jmp again
jmp again
     printf("%5d DEFICIENT\n",a);
lesser:
mov bh, value
cmp bh,9
jg greater3
mov ah,2
mov dl,value
add dl,48
int 21h
printn " DEFICIENT"
jmp again
     ;for two digit output
greater3:
mov al, value
mov ah,0
aaa
mov bx,ax
mov ah,2
mov dl,bh
add dl,48
```

int 21h

```
mov dl,bl
add dl,48
int 21h
printn " DEFICIENT"
jmp again
;jump exit
exit:
mov ah,4ch
int 21h
main endp
end main
```

UVA 382 C

```
#include<stdio.h>
int main()
  int a, sum, i;
  printf("PERFECTION OUTPUT\n");
  while(scanf("%d",&a)==1)
    sum=0;
    if(a==0){
      printf("END OF OUTPUT\n");
      return 0;
    for(i=1;i<=a/2;i++)
      if(a\%i == 0)
        sum=sum+i;
    if(sum==a)
      printf("%5d PERFECT\n",a);
    else if(sum<a)
      printf("%5d DEFICIENT\n",a);
    else
      printf("%5d ABUNDANT\n",a);
  }
  return 0;
}
```

UVA 488 Assembly

```
include "emu8086.inc"
.model small
.stack 100h
.data
case db?
limit db?
line db?
idb?
idb?
kdb?
Idb?
m db?
ndb?
.code
main proc
      ;fetching data fro data segment
  mov ax,@data
  mov ds,ax
 again:
  mov i,1
              ;for(i=1;
              ;for(j=1;
  mov j,1
  mov k.1
              ;for(k=1;
              ;for(l=1;
  mov I,1
               ;for(n=1;
  mov n,1
  mov dl,0
  call input
  mov case,dl ;scanf("%d",&t)
  printn
 case_for:
  mov bl,i
  cmp bl,case ;i<=t;
  ig exit
  mov dl,0
  call input
  mov limit,dl
                 ;scanf("%d",&a);
  printn
  mov bl,limit
  sub bl,1
  mov m,bl
               ;for(m=limit-1;
  mov dl,0
  call input
  mov line,dl
                ;scanf("%d",&b);
  printn
 main_for:
  mov bl,j
  cmp bl,line
                  ;j<=b;
  jg case_for_inc ;if j>b then case_for_inc
 in_for_1:
```

```
mov bl,k
 cmp bl,limit
                   ;k<=a;
                  ;if k>a then in for 2
 jg in for 2
in_in_for_1:
 mov bl,l
                ;l<=k;
 cmp bl,k
 ig in for 1 inc
                    ;if l>k then in for 1 inc
 mov dl,k
              ;printf("%d",k);
 add dl,48
 mov ah,2
 int 21h
 inc l
             ;|++)
 jmp in_in_for_1
in_for_1_inc:
 mov I,1
 printn
 inc k
           ;k++)
 jmp in_for_1
in_for_2:
 mov bl,m
 cmp bl,1
             ;m>=1;
 jl main_for_inc
in_in_for_2:
 mov bl.n
 cmp bl,m
               ; n<=m;
                   ;if n>m then in_for_2_dec
 jg in_for_2_dec
 mov dl,m
 add dl,48
               ;printf("%d",m);
 mov ah,2
 int 21h
 inc n
             ;n++)
jmp in_in_for_2
in_for_2_dec:
 mov n,1
 printn
 dec m
              ;m--)
 jmp in for 2
main for inc:
 mov bl,i
 cmp bl,case
                 ;if(k!=t ||
 jne new line
                  ;jmp new_line
 mov bl,j
 cmp bl,line
                 ;|| !!=b)
 jne new_line
                  ;jmp new_line
 jmp real_main_for_inc ;else
          ;jmp real_main_for_in new_line:
 printn
          ;\n
real main for inc:
 mov k,1
 mov bl,limit
```

```
sub bl,1
  mov m,bl
  inc j
              ;j++)
  jmp main_for
 case_for_inc:
  mov j,1
  inc i
             ;i++)
  jmp case_for
 exit:
  mov ah,4ch
  int 21h
               ;return 0;
  main endp
input proc
 loop_input:
  mov ah,1
                ;checking for new line
  int 21h
  cmp al,13
  je loop_exit
  sub al,48
  mov bl,al
  mov al,dl
  mov ah,0
                ;dl=1 or 2 digit input
  mov bh,10
  mul bh
  add al,bl
  mov dl,al
  jmp loop_input
 loop_exit:
  ret
  input endp
end main
```

UVA 488 C

```
#include<stdio.h>
int main()
  int t,a,b,i,j,k,l;
  while(scanf("%d",&t)==1)
    for(k=1; k<=t; k++)
       scanf("%d%d",&a,&b);
       for(l=1; l<=b; l++)
         for(j=1; j<=a; j++)
         {
            for(i=1; i<=j; i++)
              printf("%d",j);
            printf("\n");
         for(j=a-1; j>=1; j--)
            for(i=1; i<=j; i++)
              printf("%d",j);
            }
            printf("\n");
         if(k!=t || !!=b)
            printf("\n");
       }
    }
  }
  return 0;
```

| inc i inc i istack 100h idata istack 100h idb ? input db ? iob ? input db ? iocode main proc ifetching data fro data segment mov ax,@data mov ds,ax printn "this only works for 1 to 11 input because" printn "after this kind of input makes overflow to the register" again: mov input,0 itaking the input to the 2 digit loop_input: mov ah,1 int 21h cmp al,13 je loop_exit1 sub al,48 mov bl,al mov al,input mov ah,0 mov bh,10 mov bh,10 mov bh,10 mov bh,10 mov bh,10 mov ah,0 imput div bl mov sum,ah int 21h cmp al,13 je geater mov dl,input add dl,48 mov dl,input add dl,48 mov oiput,al jmp loop_input; mov ah,0 int 21h int 21h cmp al,13 je greater mov dl,input add dl,48 mov ah,2 int 21h jmp result greater: mov al,input int 21h jmp result greater: mov al,input mov al,input int 21h jmp result mov al,input greater: mov al,input | UVA 568 Assembly | mov bl,i ;sum=sum*i; |
|--|--|---|
| .model small .stack 100h .data sum db ? input db ? input db ? .code main proc | | mul bl |
| .stack 100h .data .stack 100h .data sum db ? idb ? input db ? input db ? .code main proc ;fetching data fro data segment mov ax,@data mov ds,ax printn "this only works for 1 to 11 input because" printn "fater this kind of input makes overflow to the register" again: mov input,0 ;taking the input to the 2 digit loop_input: mov ah,1 int 21h cmp al,13 je loop_exit1 sub al,48 mov bl,al mov ab,0 mov bh,10 mul bh add al,bl mov ab,0 mov bh,10 mul bh add al,bl mov input,al jmp loop_input mov al,input | include "emu8086.inc" | mov sum,al |
| loop_while: mov al,sum mov ah,0 mov bl,10 ;sum/=10; div bl cmp ah,0 jg loop_for jetching data fro data segment mov ax, ax printn "this only works for 1 to 11 input because" printn "after this kind of input makes overflow to the register" again: mov input,0 ;taking the input to the 2 digit loop_input: mov ah,1 int 21h cmp al,13 je loop_exit1 | .model small | inc i |
| sum db? idb? idb? code main proc jeftching data fro data segment mov ax,@data mov ds,ax printn "this only works for 1 to 11 input because" printn "after this kind of input makes overflow to the register" again: mov input,0 jtaking the input to the 2 digit loop_input: mov ah,1 int 21h cmp al,13 je loop_exit1 sub al,48 mov bl,al mov al,input mov ah,0 mov bh,10 mul bh add al,bl mov al,input mov al,input cmp al,0 je loop_input mov li,1 jin loop_input mov al,input cmp al,0 je loop_input mov al,input cmp al,0 je loop_input mov sum,1 mov i,1 jfor(i=1;i<=a;i++) loop_exit mov al,input mov a | .stack 100h | ;while(sum%10==0) |
| ind b? input db? code main proc ;fetching data fro data segment mov ax,@data mov ds,ax printn "this only works for 1 to 11 input because" printn "after this kind of input makes overflow to the register" again: mov input,0 ;taking the input to the 2 digit loop_input: mov ah,1 int 21h cmp al,13 je loop_exit1 sub al,48 mov bl,al mov al,10 mov ah,0 mov bh,10 mov sum,al jmp loop_for loop_exit: printn mov al,sum mov al,sum mov ah,0 inv bl,10 ;sum=sum%10; div bl mov sum,ah int 21h cmp al,13 je loop_exit1 mov al,input mov ah,0 mov bb,10 mov injut,al jmp loop_input jafter taking input checking whether the input value is null loop_exit: printn mov al,sum mov bl,10 ;sum=sum%10; div bl mov sum,ah inv bl,input cmp bl,9 jg greater mov dl,input add dl,48 mov ah,2 jfor 1 digit output mov ah,0 aaa mov ah,0 abd dl,48 for 2 digit output mov ah,2 int 21h loop_for: mov al,input mov al,input mov al,input mov ah,0 aaa mov bx,ax mov al,input mov ah,0 aaa mov bx,ax int 21h loop_for: mov al,input mov al, | .data | loop_while: |
| input db? .code .c | sum db? | mov al,sum |
| div bl cmp ah,0 jg loop_for jfetching data fro data segment mov ax,@data mov ds,ax printn "this only works for 1 to 11 input because" printn "after this kind of input makes overflow to the register" again: mov input,0 jtaking the input to the 2 digit loop_input: mov ah,1 int 21h cmp al,13 je loop_exit1 sub al,48 mov bl,al mov al,1pnut mov ah,0 mov bh,10 mov h,0 mov bh,10 mov bh,10 mov h,0 mov bh,10 mov l,1 jmp loop_exit printn mov al,sum mov al,0 je loop_input je loop_input jafter takinng input checking whether the input value is null loop_exit: mov al,input mov ah,0 loop_input je loop_input je loop_input mov al,input mov al,i | idb? | mov ah,0 |
| main proc ; fetching data fro data segment mov ax,@data mov dx,ax printn "this only works for 1 to 11 input because" printn "after this kind of input makes overflow to the register" again: mov input,0 ; taking the input to the 2 digit loop_input mov ah,1 int 21h cmp al,13 je loop_exit1 sub al,48 mov bl,al mov al,input mov ah,0 mov bh,10 mov bh,10 mov bh,10 mov lainput mov ah,0 mov bh,10 mov lainput mov ah,0 mov bh,10 mov lainput jimp loop_input jafter takinng input checking whether the input value is null loop_exit1: mov al,input | input db? | mov bl,10 ;sum/=10; |
| ;fetching data fro data segment mov ax,@data mov dx,ax jmp loop_while jmp loop_for because" loop_exit: printn again: mov input,0 mov ah,1 mov ah,1 mov ah,1 int 21h mov ah,1 int 21h mov al,13 jp loop_exit mov al,148 mov bl,18 mov bl,19 jg greater mov bl,10 mov add dl,48 mov bh,10 mov input,al jmp loop_input aaa aa mov bl,10 mov linput mov al,input mov al,input mov al,input mov al,input mov al,input mov sum,1 mov i,1 jfor(i=1;i<=a;i++) loop_exit mov al,input mov dl,input add dl,48 mov dl,19 int 21h loop_for: mov al,input mov al,input mov ah,2 int 21h loop_for: mov al,input | .code | div bl |
| mov ax,@data mov ds,ax printn "this only works for 1 to 11 input because" printn "after this kind of input makes overflow to the register" again: mov input,0 | main proc | cmp ah,0 |
| mov ax,@data mov ds,ax printn "this only works for 1 to 11 input because" printn "after this kind of input makes overflow to the register" again: mov input,0 | ;fetching data fro data segment | jg loop_for ;if ==0 then again go to for loop |
| mov ds,ax printn "this only works for 1 to 11 input because" printn "after this kind of input makes overflow to the register" again: mov input,0 ;taking the input to the 2 digit loop_input: mov ah,1 int 21h cmp al,13 je loop_exit1 sub al,48 mov bl,al mov ah,0 mov bh,10 mov ah,0 mov bh,10 mov ah,0 mov bh,10 mul bh add al,bl mov input,al jmp loop_input ;after takinng input checking whether the input value is null loop_exit1: mov al,input cmp al,0 je loop_input mov sum,1 mov i,1 jfor(i=1;i<=a;i++) loop_for: mov dl,input mov dl,input mov dl,input mov ah,0 aaa mov bx,ax jmp loop_exit: printf mov al,sum mov al,sum mov ah,0 mov bl,10 ;sum=sum%10; loop_exit0; sum=sum%10; loop_exit1 mov sum,ah imov sum,ah imov bl,i0 jg greater mov dl,input mov ah,2 ;for 1 digit output int 21h jmp result mov ah,0 aaa mov bx,ax mov bx,ax mov dl,bh add dl,48 jfor 2 digit output mov ah,2 int 21h loop_for: mov dl,bl add dl,48 mov dl,bl add dl,48 mov dl,bl add dl,48 ij loop_exit mov dl,bl add dl,48 mov dl,2 int 21h add dl,48 int 21h add dl, | | mov sum,al |
| because" printn "after this kind of input makes overflow to the register" again: mov input,0 ;taking the input to the 2 digit loop_input: mov ah,1 int 21h cmp al,13 je loop_exit1 sub al,48 mov bl,al mov al,input mov ah,0 mov bh,10 mov la,input mov input,al jmp loop_input; ;after takinng input checking whether the input value is null loop_exit1: mov al,input mov al,input mov al,input mov sum,1 mov sum,1 mov al,input mov al,input mov al,input mov bl,al mov sum,ah int 21h jfor 1 digit output mov al,input mov al,input mov al,input mov al,input mov al,input mov bl,al mov bl,al mov al,input mov al,input mov al,input mov al,input mov sum,1 mov i1 int 21h add dl,48 mov bl,al mov al,input mov al,2 int 21h loop_for: mov al,input mov al,2 int 21h loop_exit mov al,48 mov al,60 mov | | |
| to the register" again: mov input,0 ;taking the input to the 2 digit loop_input: mov ah,1 int 21h cmp al,13 je loop_exit1 mov al,19 sub al,48 mov bl,al mov ab,0 mov bh,10 mov ah,0 mov bh,10 mov bh,10 mul bh add al,bl mov input,al jmp loop_input ;after takinng input checking whether the input value is null loop_exit1: mov al,input alad al,48 mov al,2 int 21h alad a | printn "this only works for 1 to 11 input because" | jmp loop_for |
| again: | printn "after this kind of input makes overflow | loop_exit: |
| mov input,0 mov ah,0 loop_input: mov ah,1 mov ah,1 mov sum,ah int 21h printf("%5d -> %lld\n",a,sum); cmp al,13 printf("%5d -> %lld\n",a,sum); sub al,48 pig greater mov bl,al mov dl,input mov ah,0 mov dl,input mov ah,0 mov dl,input mov ah,0 add dl,48 mov input,al mov ah,2 for 1 digit output int 21h int 21h mov ah,0 add al,bl mov al,input mov ah,0 aaa mov ah,0 aaa input value is null mov al,input mov ah,0 loop_exit1: mov al,input mov dl,bh mov al,input mov dl,bh add dl,48 mov al,input mov al,2 int 21h loop_for: mov al,2 int 21h loop_for: mov al,input mov dl,bl add dl,48 mov al,2 int 21h loop_for: mov al,0 add dl,48 mov al,input ad | to the register" | printn |
| ;taking the input to the 2 digit loop_input: mov ah,1 int 21h cmp al,13 je loop_exit1 mov bl,al mov bl,al mov ah,0 mov bh,10 mov bh,10 mov input,al jmp loop_input ;after taking input checking whether the input value is null loop_exit1: mov al,input cmp al,0 je loop_input mov sum,1 mov i,1 jfor(i=1;i<=a;i++) loop_for: mov al,1 int 21h div bl mov sum,ah int 21h mov sum,ah int 21h jig greater mov dl,input add dl,48 mov ah,2 jfor 1 digit output int 21h jmp result mov ah,0 aaa mov bx,ax cmp al,0 je loop_input mov sum,1 mov i,1 jfor(i=1;i<=a;i++) loop_for: mov al,input cmp al,i jl loop_exit mov ah,2 idv bl mov sum,ah int 21h jmp result mov ah,2 jfor 1 digit output mov al,input mov al,input mov al,input mov dl,bh add dl,48 mov dl,bl add dl,48 mov ah,2 | again: | mov al,sum |
| loop_input: mov ah,1 int 21h cmp al,13 je loop_exit1 sub al,48 mov bl,al mov ah,0 mov ah,0 mov input,al jmp loop_input jafter takinng input checking whether the input value is null loop_exit1: mov al,input cmp al,0 je loop_input int 21h add al,bl mov al,input add dl,48 mov al,2 int 21h loop_exit | mov input,0 | mov ah,0 |
| mov ah,1 int 21h cmp al,13 je loop_exit1 sub al,48 mov bl,al mov ah,0 mov bh,10 mul bh add al,bl mov input,al jmp loop_input ;after takinng input checking whether the input value is null loop_exit1: mov al,input mov al,0 je loop_input mov al,input mov al,input mov al,input mov al,input mov al,input mov al,input mov bx,ax mov al,input mov bx,ax mov dl,bh add dl,48 mov ah,0 aaa mov bx,ax mov dl,bh add dl,48 jfor 2 digit output mov ah,2 int 21h mov ah,0 je loop_for: mov al,input mov ah,0 je loop_for: mov al,input mov bx,ax mov dl,bh add dl,48 mov dl,bl add dl,48 mov ah,2 int 21h mov al,input mov al,input mov ah,2 int 21h mov al,2 int 21h mov al,2 int 21h mov al,48 mov al,19 int 21h i | taking the input to the 2 digit; | mov bl,10 ;sum=sum%10; |
| int 21h cmp al,13 je loop_exit1 sub al,48 mov bl,al mov al,input mov ah,0 mov bh,10 mul bh add al,bl mov input,al jmp loop_input ;after takinng input checking whether the input value is null loop_exit1: mov al,10 je loop_input mov sum,1 mov sum,1 mov sum,1 mov i,1 jfor(i=1;i<=a;i++) loop_for: mov al,input cmp al,0 je loop_exit mov al,input mov ah,0 add dl,48 jfor 2 digit output mov dl,bl add dl,48 jfl loop_exit mov dl,bl add dl,48 mov oh,2 int 21h mov dl,bl add dl,48 jfl loop_exit mov dl,bl add dl,48 jfl loop_exit mov dl,bl add dl,48 mov ah,2 int 21h | loop_input: | div bl |
| cmp al,13 ;printf("%5d -> %lld\n",a,sum); je loop_exit1 mov bl,input sub al,48 jg greater mov bl,al mov dl,input mov ah,0 add dl,48 mov bh,10 mov ah,2 ;for 1 digit output mul bh int 21h add al,bl jmp result mov input,al jmp result ipm loop_input greater: input value is null mov al,input loop_exit1: mov al,input mov al,input mov bx,ax cmp al,0 mov dl,bh je loop_input mov dl,bh mov sum,1 mov al,input mov al,input mov al,input int 21h int 21h loop_for: mov al,input mov al,input mov al,input mov al,input mov al,input mov al,input mov al,input mov al,input mov al,input aaa mov al,input aba mov al,input aaa mov al,input aba int 21h loop_exit mov al,input | mov ah,1 | mov sum,ah |
| je loop_exit1 sub al,48 mov bl,al mov al,input mov ah,0 mov bl,10 mul bh add al,bl mov input,al jmp loop_input input value is null loop_exit1: mov al,input mov sum,1 mov bl,input cmp bl,9 jg greater mov dl,input add dl,48 mov ah,2 jfor 1 digit output int 21h jmp result mov al,input mov al,input mov ah,0 aaa mov bx,ax cmp al,0 je loop_input mov sum,1 mov i,1 jfor(i=1;i<=a;i++) loop_for: mov al,input cmp al,i jl loop_exit mov bl,input mov bl,input mov ah,0 aaa mov bx,ax cmp al,0 je loop_input mov sum,1 mov ah,2 jfor(i=1;i<=a;i++) loop_for: mov al,input cmp al,i jl loop_exit mov dl,bl add dl,48 mov ah,2 | int 21h | |
| sub al,48 jg greater mov bl,al mov dl,input mov al,input mov dl,input mov bh,10 mov ah,2 ;for 1 digit output mul bh int 21h add al,bl jmp result mov input,al jmp result jmp loop_input greater: ;after takinng input checking whether the mov al,input input value is null mov ah,0 loop_exit1: aaa mov al,input mov bx,ax cmp al,0 mov dl,bh je loop_input mov dl,bh mov al,input mov ah,2 jfor(i=1;i<=a;i++) | cmp al,13 | ;printf("%5d -> %lld\n",a,sum); |
| sub al,48 mov bl,al mov al,input mov ah,0 mov bh,10 mul bh add al,bl mov input,al jmp loop_input input value is null loop_exit1: mov al,input mov al,input mov ab,0 gloop_exit4: mov al,input mov al,2 int 21h mov al,2 int 21h mov al,2 int 21h mov al,48 mov al,2 | je loop_exit1 | mov bl,input |
| mov bl,al mov al,input mov ah,0 mov bh,10 mov bh,10 mul bh add al,bl mov input,al jmp loop_input ;after takinng input checking whether the input value is null loop_exit1: mov al,input cmp al,0 je loop_input mov sum,1 mov al,input mov al,1 ;for(i=1;i<=a;i++) loop_for: mov al,input mov al,input mov ah,2 jl loop_exit mov al,input mov ah,2 int 21h loop_for: mov al,input mov dl,bl add dl,48 jl loop_exit mov dl,48 jl loop_exit mov ah,2 int 21h mov dl,bl add dl,48 jl loop_exit mov ah,2 int 21h | | cmp bl,9 |
| mov al,input mov ah,0 mov bh,10 mul bh add al,bl mov input,al jmp loop_input input value is null loop_exit1: mov al,input mov al,input mov al,input mov al,input mov al,input mov i,1 jfor(i=1;i<=a;i++) loop_for: mov al,input mov al,input mov al,input mov al,input mov al,2 jfor(i=1;i<=a;i++) loop_exit mov al,input mov dl,bh add dl,48 jfor 2 digit output mov al,2 jint 21h loop_for: mov al,input mov dl,bl add dl,48 jil loop_exit mov al,2 jil oop_exit mov al,2 jil oop_exit mov al,2 jil oop_exit mov al,2 jil oop_exit mov al,2 mov al,2 mov dl,bl add dl,48 mov al,2 | sub al,48 | jg greater |
| mov ah,0add dl,48mov bh,10mov ah,2;for 1 digit outputmul bhint 21hadd al,bljmp resultmov input,algreater:jmp loop_inputgreater:;after takinng input checking whether themov al,inputinput value is nullmov ah,0loop_exit1:aaamov al,inputmov bx,axcmp al,0mov dl,bhje loop_inputmov dl,bhmov sum,1add dl,48mov i,1int 21hifor(i=1;i<=a;i++) | mov bl,al | |
| mov bh,10mov ah,2;for 1 digit outputmul bhint 21hadd al,bljmp resultmov input,aljmp resultjmp loop_inputgreater:;after takinng input checking whether themov al,inputinput value is nullmov ah,0loop_exit1:aaamov al,inputmov bx,axcmp al,0mov dl,bhje loop_inputmov dl,48mov sum,1add dl,48jfor(i=1;i<=a;i++) | mov al,input | mov dl,input |
| mul bh add al,bl mov input,al jmp loop_input ;after takinng input checking whether the input value is nullint 21h jmp resultloop_exit1: mov al,input cmp al,0 je loop_input mov sum,1 mov i,1 ;for(i=1;i<=a;i++) | mov ah,0 | add dl,48 |
| add al,bl mov input,al jmp loop_input ;after takinng input checking whether the input value is null loop_exit1: mov al,input cmp al,0 je loop_input mov sum,1 mov i,1 mov i,1 jfor(i=1;i<=a;i++) loop_for: mov al,input mov dl,bl cmp al,i jl loop_exit jmp result mov al,input mov al,input mov ah,0 aaa mov bx,ax mov dl,bh mov dl,bh mov ah,2 int 21h mov dl,bl add dl,48 jl loop_exit mov ah,2 | mov bh,10 | mov ah,2 ;for 1 digit output |
| mov input,al jmp loop_input ;after takinng input checking whether the input value is null loop_exit1: mov al,input cmp al,0 je loop_input mov i,1 jfor(i=1;i<=a;i++) loop_for: mov al,input cmp al,i jl loop_exit greater: mov al,input mov al,input mov bx,ax mov bx,ax mov dl,bh mov dl,bh mov ah,2 int 21h mov dl,bl add dl,48 jl loop_exit mov ah,2 | mul bh | int 21h |
| jmp loop_input ;after takinng input checking whether the input value is null loop_exit1: mov al,input cmp al,0 je loop_input mov sum,1 mov i,1 jfor(i=1;i<=a;i++) loop_for: mov al,input mov dl,bl mov ah,2 int 21h greater: mov al,input mov ah,0 aaa mov bx,ax mov bx,ax for 2 digit output mov ah,2 int 21h loop_for: mov al,input mov dl,bl add dl,48 jl loop_exit mov ah,2 | add al,bl | jmp result |
| ;after takinng input checking whether the input value is null mov ah,0 loop_exit1: aaa mov bx,ax cmp al,0 je loop_input mov sum,1 mov sum,1 add dl,48 ;for 2 digit output mov ah,2 ;for(i=1;i<=a;i++) int 21h loop_for: mov al,input mov dl,bl cmp al,i add dl,48 jl loop_exit mov ah,2 | mov input,al | |
| input value is null loop_exit1: mov al,input cmp al,0 je loop_input mov dl,bh mov sum,1 mov ah,2 ;for(i=1;i<=a;i++) loop_for: mov al,input cmp al,i jl loop_exit mov ah,0 aaa mov ah,0 ada mov bx,ax mov dl,bh add dl,48 ;for 2 digit output mov ah,2 int 21h mov dl,bl add dl,48 jl loop_exit mov ah,2 | jmp loop_input | greater: |
| loop_exit1: aaa mov al,input mov bx,ax cmp al,0 mov dl,bh je loop_input mov dl,bh mov sum,1 add dl,48 ;for 2 digit output mov al,2 int 21h loop_for: mov dl,bl cmp al,i add dl,48 jl loop_exit mov ah,2 | ;after takinng input checking whether the | mov al,input |
| mov al,input mov bx,ax cmp al,0 je loop_input mov dl,bh mov sum,1 add dl,48 ;for 2 digit output mov i,1 mov ah,2 ;for(i=1;i<=a;i++) int 21h loop_for: mov al,input mov dl,bl cmp al,i add dl,48 jl loop_exit mov ah,2 | input value is null | mov ah,0 |
| cmp al,0 je loop_input mov dl,bh mov sum,1 add dl,48 ;for 2 digit output mov i,1 mov ah,2 ;for(i=1;i<=a;i++) | loop_exit1: | aaa |
| je loop_input mov dl,bh mov sum,1 add dl,48 ;for 2 digit output mov i,1 mov ah,2 ;for(i=1;i<=a;i++) int 21h loop_for: mov al,input mov dl,bl cmp al,i add dl,48 jl loop_exit mov ah,2 | mov al,input | mov bx,ax |
| mov sum,1 add dl,48 ;for 2 digit output mov i,1 mov ah,2 ;for(i=1;i<=a;i++) | cmp al,0 | |
| mov i,1 mov ah,2 ;for(i=1;i<=a;i++) | je loop_input | mov dl,bh |
| ;for(i=1;i<=a;i++) int 21h loop_for: mov al,input mov dl,bl cmp al,i add dl,48 jl loop_exit mov ah,2 | mov sum,1 | add dl,48 ;for 2 digit output |
| loop_for: mov al,input mov dl,bl cmp al,i add dl,48 jl loop_exit mov ah,2 | mov i,1 | mov ah,2 |
| mov al,input mov dl,bl cmp al,i add dl,48 jl loop_exit mov ah,2 | ;for(i=1;i<=a;i++) | int 21h |
| mov al,input mov dl,bl cmp al,i add dl,48 jl loop_exit mov ah,2 | | |
| cmp al,i add dl,48 jl loop_exit mov ah,2 | · - | mov dl,bl |
| jl loop_exit mov ah,2 | • | |
| | | |
| | · · · · · — | |

```
result:
  print " -> "
  mov bl,sum
  cmp bl,9
  jg greater2
  mov dl,sum
  add dl,48
  mov ah,2
                ;for 1 digit output
  int 21h
  jmp restart
 greater2:
  mov al, sum
  mov ah,0
  aaa
  mov bx,ax
  mov dl,bh
  add dl,48
                   ;for 2 digit output
  mov ah,2
  int 21h
  mov dl,bl
  add dl,48
  mov ah,2
  int 21h
 restart:
  printn
  jmp again
 exit:
  mov ah,4ch
  int 21h
               ;return 0;
  main endp
end main
```

UVA 568 C

```
#include<stdio.h>
int main()
{
  int a,i;
  long long sum;
  while(scanf("%d",&a)==1)
  {
    sum=1;
    for(i=1;i<=a;i++)
      sum=sum*i;
      while(sum%10==0)
        sum/=10;
        sum%=100000;
    }
    sum=sum%10;
    printf("%5d -> %lld\n",a,sum);
 }
  return 0;
}
```

UVA 591 Assembly

```
include "emu8086.inc"
.model small
.stack 100h
.data
t dw?
sum dw?
sum2 dw?
idw?
n dw?
m dw?
j dw?
value dw?
count dw?
a dw 100 dup(0)
.code
main proc
     ;fetching all data
  mov ax,@data
  mov ds,ax
again:
  mov j,1
  mov count,0
                ;count=0
  mov value,0
  call input
  mov ax, value
                ;scanf("%d",&t)!=
  mov t,ax
  printn
  cmp ax,0
  je exit
          ;checking whetther t!=0
  mov sum,0 ;sum=0;
  mov sum2,0 ;sum2=0;
  mov i,0 ;for(i=0;
  mov si,0 ; array index
 for1:
  mov ax,i
  cmp ax,t
  jge for2_zero ;i<T;
  mov value,0
  call input
             ;scanf("%d",&a[i]);
  mov ax, value
```

```
printn
 mov a[si],ax
 add ax,sum
 mov sum,ax
                ;sum+=a[i];
 inc si; as define word then index+=2
 inc si
 inc i ;i++)
jmp for1
for2_zero:
 xor dx,dx
 mov ax,sum
 mov bx,t
 div bx
 mov n,ax ;n=sum/t;
 mov i,0 ;for(i=0;
 mov si,0 ;array index
for2:
 mov ax,i
 cmp ax,t
 jge push_before ;i<T;
 mov ax,n
 cmp a[si],ax
 jle increament ;if(a[i]>n)
 mov ax,a[si]
 sub ax,n
 mov m,ax
            ; m=a[i]-n;
 mov ax,m
 add sum2,ax ;sum2+=m;
increament:
       ; as define word then index+=2
 inc si
 inc si
 inc i ;i++)
jmp for2
push before:
 print "Set#"
               ;printf("Set #
push_value:
 mov ax,j
           ;checking for j is 0 or not
 cmp ax,0
 je pop_value
 xor dx,dx
 mov bx,10
           ;sum/10
```

div bx

```
push dx
             ;pushing last digit as reminder
                                                          jmp pop_value2
  mov j,ax
                                                          exit2:
                                                           printn
  inc count
              ;value length increase
                                                          inc j ;j++;
                                                          jmp again ;calliing the program again
  jmp push_value
                                                           mov ah,4ch
                                                          int 21h
 pop_value:
  mov ax,count
  cmp ax,0
              ;checking for value length
                                                           main endp
  je push before2
                                                        input proc
  dec count
                                                           push ax
                                                           push bx
  pop dx
                                                           push cx ;saving all data if used
  add dx,48
                                                           push dx
             ;printing digit from stack
  mov ah,2
  int 21h
                                                         for_loop:
                                                           mov ah,1
  jmp pop_value
                                                          int 21h
                                                                     ;getting input
                                                           cbw
 push_before2:
                                                           cmp ax,13
             ;printf("The minimum number of
                                                          je exit3 ;checking whether it is new line
  printn
moves is")
  print "The minimum number of moves is "
                                                          cmp ax,32
                                                          je exit3 ;checking whether it is space
 push value2:
  mov ax,sum2
  cmp ax,0 ;checking for sum2 is 0 or not
                                                           sub ax,48 ;making pure digit
  je pop_value2
                                                           mov cx,ax ;cx=input
  xor dx,dx
  mov bx,10
                                                           mov ax, value
  div bx
            ;sum/10
                                                           mov bx,10 ;value=value*10
                                                           mul bx
  push dx
            ;pushing last digit as reminder
  mov sum2,ax
                                                           add ax,cx ;recent result+input
              ;value length increase
                                                           mov value,ax ;value=recent result
  inc count
  jmp push_value2
                                                          jmp for_loop ;loop call
 pop_value2:
                                                         exit3:
  mov ax, count
                                                          pop dx
              ;checking for value length
  cmp ax,0
                                                           рор сх
  je exit2
                                                           pop bx
                                                                   ;restoring all registor value
  dec count
                                                           pop ax
                                                           ret
  pop dx
                                                          input endp
  add dx,48
                                                        end main
  mov ah,2
             ;printing digit from stack
```

int 21h

UVA 591 C

```
#include<stdio.h>
int main()
  int T,sum,sum2,i,n,m,j=1;
  while(scanf("%d",&T)==1 && T!=0)
    sum=0;
    sum2=0;
    int a[T];
    for(i=0; i<T; i++)
      scanf("%d",&a[i]);
      sum+=a[i];
    n=sum/T;
    for(i=0; i<T; i++)
    {
      if(a[i]>n)
        m=a[i]-n;
        sum2+=m;
      }
    printf("Set #%d\nThe minimum number of
moves is %d.n\n",j,sum2);
    j++;
  return 0;
```

```
UVA 900 Assembly
                                                            printn
                                                            jmp again
include "emu8086.inc"
.model small
                                                           greater:
.stack 100h
                                                            mov al,b
.data
                                                            mov ah,0
testcase db?
                                                            aaa
adb?
                                                            mov bx,ax
bdb?
                                                                               ;printf("%d\n",b); for 2 digit
                                                            mov dl,bh
.code
                                                            add dl,48
main proc
                                                            mov ah,2
                                                            int 21h
      ;fetching data fro data segment
  mov ax,@data
                                                            mov dl,bl
  mov ds,ax
                                                            add dl,48
                                                            int 21h
 again:
                                                            printn
    ;making the value set
                                                            jmp again
  mov a,0
  mov b,1
                                                           exit:
                                                            mov ah,4ch
      ;taking the testcase input while
                                                            int 21h
(testcase!=0)
                                                            main endp
  mov ah,1
                                                          end main
  int 21h
  sub al.48
  cmp al,0
  je exit
  mov testcase, al
  mov al,0
               ;for(int i=0
 for start:
                                                          UVA 900 C
  cmp al,testcase ;i<testcase
  jge result
  inc al
             ;i++
                                                          #include<stdio.h>
                                                          int main()
  mov bh,b ;c=b;
                                                          {
  mov bl,a
                                                            int t,i;
  add bl,bh ;b=a+b;
                                                            while(scanf("%d",&t)==1 && t!=0)
  mov b,bl
                                                              int a=0,b=1,sum=0,c;
             ;assign to b
                                                              for(i=0;i<t;i++)
  mov a,bh
               ;a=c;
  jmp for_start
                                                                c=b;
 result:
                                                                b=a+b;
  printn
                                                                a=c;
  mov bl,b
  cmp bl,9
                 ;printf("%d\n",b); for 1 digit
                                                              printf("%d\n",b);
  jg greater
  mov dl,b
                                                            return 0;
  add dl,48
  mov ah,2
```

int 21h

| UVA 913 Assembly | inc k ;k++) |
|--|--|
| | jmp for2 |
| include "emu8086.inc" | print_sum: |
| .model small | mov dl,sum |
| .stack 100h | cmp dl,9 |
| .data | jg greater |
| input_value db ? | add dl,48 |
| i db ? | mov ah,2 ;printf("%lld\n",sum); for 1 digit |
| j db ? | output |
| k db? | int 21h |
| sum db? | jmp jump_again |
| final db? | greater: |
| .code | mov al,sum |
| main proc | mov ah,0 |
| mov ax,@data | mov bl,10 |
| mov ds,ax | div bl |
| again: | mov bx,ax |
| mov dl,0 | mov dl,bl |
| call input | mov ah,2 |
| mov input value,dl ;scanf("%lld",&n) | add dl,48 ;printf("%lld\n",sum); for 2 digit |
| printn | output |
| mov j,1 ;j=1 | int 21h |
| mov sum,0 ;sum=0 | mov dl,bh |
| mov i,1 ;for(i=1; | add dl,48 |
| for1: | int 21h |
| mov bl,i | jump_again: |
| cmp bl,input_value ;i <input_value;< td=""><td>printn</td></input_value;<> | printn |
| jge final_get | jmp again |
| inc j ;j++ | exit: |
| inc i ;i+=2) | mov ah,4ch |
| inc i | int 21h |
| jmp for1 | main endp |
| final_get: | input proc |
| ;final=(j*i)+j-1; | loop_input: |
| mov al,j ;al=j | mov ah,1 |
| mov ah,0 ;ah=0 | int 21h ;checking for new line |
| mov bl,i ;bl=i | cmp al,13 |
| mul bl ;al=al*bl | je loop_exit |
| add al,j ;al+=j | sub al,48 |
| sub al,1 ;al-=1 | mov bl,al |
| mov final,al ;final=al | mov al,dl |
| mov k,1 ;for(k=1; | mov ah,0 ;dl=1 or 2 digit input |
| for2: | mov bh,10 |
| mov bl,k | mul bh |
| cmp bl,3 ;k<=3; | add al,bl |
| • | mov dl,al |
| jg print_sum | |
| mov al,sum add al,final | jmp loop_input |
| | loop_exit: ret |
| mov sum,al ;sum=sum+final; | |
| dec final ;final-=2; | input endp |
| dec final | end main |

UVA 913 C

```
#include<stdio.h>
int main()
  long long int n;
  while(scanf("%lld",&n)==1)
  {
    long long int i,j=1,finall,sum=0,k;
    for(i=1;i<n;i+=2)
      j++;
    finall=(j*i)+j-1;
    for(k=1;k<=3;k++)
      sum=sum+finall;
      finall-=2;
    printf("%lld\n",sum);
  }
  return 0;
}
```

UVA 1124 Assembly

```
include "emu8086.inc"
.model small
.stack 100h
.data
str1 db 255
.code
main proc
  lea si,str1
 input:
  mov ah,1
  int 21h
  cmp al,13
                ;gets(a)
  je print
  mov [si],al
  inc si
  jmp input
print:
  printn
  inc si
  mov dl,'$'
               ;setting a last finish indicator in the
last of the string
  mov [si],dl
  lea dx,str1
                 ;printf("%s\n",a);
  mov ah,9
  int 21h
  printn
  jmp input
  main endp
end main
```

UVA 1124 C

```
#include<stdio.h>
#include<string.h>
int main()
{
    char a[100000];
    while(gets(a))
    {
        printf("%s\n",a);
    }
    return 0;
}
```

```
UVA 10035 Assembly
                                                           main_for:
                                                            mov ax, value1
include "emu8086.inc"
                                                            cmp ax,0
.model small
                                                            je second_zero2
.stack 100h
                                                            jmp work
.data
value1 dw?
                                                          second_zero2:
value2 dw?
                                                            mov ax,value2
                                                                                ;if (value1==0 &&
idw?
                                                         value2==0)
carry dw?
                                                            cmp ax,0
                                                                                ;then go for printing result
r dw?
                                                            je print_result
c dw?
value dw?
                                                          work:
                                                            mov c,0
.code
main proc
                                                            mov ax,i
                                                                           ;for(i=1; i<16;
      ;fetching the data
                                                            cmp ax,16
  mov ax,@data
                                                            jge print_result
  mov ds,ax
                                                            mov ax,r
 restart:
                                                            add c,ax
                                                                            ;c+=r;
       ; calling for first multidigit number
  mov value,0
                                                            xor dx,dx
  call input
             ;input function call
                                                            mov ax, value1
  mov dx, value
                                                            mov bx,10
  mov value1,dx
                                                            div bx
                     ;value1
            ;new line
                                                            add c,dx
                                                                        ;c+= (value1%10)
  printn
       ; calling for 2nd multidigit number
  mov value,0
                                                            xor dx,dx
                                                            mov ax, value 2
  call input
                                                            mov bx,10
  mov dx, value
  mov value2.dx
                                                            div bx
  mov ax, value1
                                                            add c,dx
                                                                        ;c+=(value2%10)
  cmp ax,0
                  ;if (value1==0
  je second zero
                     ;then go for second value
                                                            mov ax,c
checking
                                                            cmp ax,9
                                                                         ;if(c>9)
  jmp main_work
                                                            jg r_1
                                                                         ;then go to r_1
 second zero:
                                                            mov r,0
                                                                          ;else r=0
  mov ax, value 2
                                                            jmp division_values
                 ;&& value2==0)
  cmp ax,0
  je exit
                 ;then jump exit
                                                           r_1:
                                                            inc carry
                                                                         ;carry++;
 main_work:
                                                            mov r,1
                                                                           ;r=1;
  mov r,0
                                                           division values:
  mov carry,0
                 ;carry=0;
                                                            xor dx,dx
             ;for(i=1;
                                                            mov ax, value1
  mov i,1
```

```
mov bx,10
  div bx
  mov value1,ax
                    ;value1/=10;
  xor dx,dx
  mov ax, value 2
  mov bx,10
  div bx
  mov value2,ax
                    ;value2/=10;
  inc i
           ;i++)
  jmp main_for
 print result:
  printn
  mov ax, carry
                ;if(carry==0)
  cmp ax,0
  je no_carry
  mov ax, carry
                 ;else if(carry==1)
  cmp ax,1
  je one_carry
  mov dx,carry ;else
  add dl,48
               ;print carry value
  mov ah,2
  int 21h
  printn " carry operation." ;printf("%d carry
operations.\n",carry);
  jmp restart
 no_carry:
  printn "No carry operation." ;printf("No carry
operation.\n");
  jmp restart
 one_carry:
  printn "1 carry operation." ;printf("1 carry
operation.\n");
  jmp restart
 exit:
  mov ah,4ch
  int 21h
  main endp
input proc
  push ax
  push bx
            ;taking all register in stack
```

```
push cx
  push dx
 input_for:
  mov ah,1
  int 21h
            ;scanf
  cmp al,13
  je exit_for
  sub al,48
  cbw
            ;converting byte to word
  mov cx,ax
  mov ax, value
  mov bx,10
               ;saving value
  mul bx
  add ax,cx
  mov value,ax ;value=value*10 + scanf
  jmp input_for
 exit_for:
  pop ax
  pop bx
             ;restoring all register
  рор сх
  pop dx
  ret
  input endp
end main
```

UVA 10035 C

```
#include<stdio.h>
int main()
{
  long int a,b;
  int i,r,c,carry;
  while(scanf("%ld%ld",&a,&b)==2)
  {
    if (a==0 && b==0)
      break;
    else
    {
      r=0;
      carry=0;
      for(i=1; i<16; i++)
         if (a==0 && b==0)
           break;
         c=(a%10)+(b%10)+r;
         if(c>9)
         {
           carry++;
           r=1;
         }
         else
         {
           r=0;
         a/=10;
         b/=10;
      }
      if(carry==0)
         printf("No carry operation.\n");
      else if(carry==1)
         printf("1 carry operation.\n");
      else
         printf("%d carry operations.\n",carry);
    }
  }
}
```

```
UVA 10055 Assembly
                                                              ;moving the data from al to number2
                                                         variable
org 100h
                                                           mov number2,al
.model small
.stack 100h
                                                              ;creating a new line
.data
                                                           mov ah,2
number1 db?
                                                           mov dl,10
number2 db?
                                                           int 21h
number3 db?
                                                           mov dl,13
hint1 db "enter the two number:$"
                                                           int 21h
hint2 db "the result is: $"
                                                           mov al,number2 ;moving the number2 value
.code
                                                         to the al
                                                           mov cl,number1 ;moving the number1 value
main proc
    ;loading all data to the data segment
                                                         to the cl
                                                           cmp cl,al
  mov ax,@data
  mov ds,ax
                                                           jg exchange
                                                                          ;if first is big than 2nd then
                                                         exchange label called
  new:
                                                                        ;else continue
   ; displaying the program hint what it is about
  mov ah.9
                                                           mov al,0
                                                                            ;moving 0 to the al registor
  lea dx,hint1
                                                           add al,number2
                                                                                ;add number2 to the al as
  int 21h
                                                         al=al+number2
                                                                                ;then subtract number1
                                                           sub al, number 1
    ;printing a new line
                                                         from al that is al=number2-number1
                                                                            ;making the ascii value from
  mov ah,2
                                                           add al,48
  mov dl,10
                                                         the pure value
  int 21h
                                                           mov number2,al
                                                                                 ;moving the value of al to
  mov dl,13
                                                         number2
  int 21h
                                                           print:
                                                             ;initialize the print label
    ;getting user input
  mov ah,1
                                                               ;printing result message
  int 21h
                                                           mov ah,9
                                                           lea dx,hint2
    ;moving the data from al to number1
                                                           int 21h
variable
  mov number1,al
                                                               ;printing the number2 that means the result
                                                           mov dl,number2
    ;creating a new line
                                                           mov ah,2
  mov ah,2
                                                           int 21h
  mov dl,10
                                                              ;prtinting a new line
  int 21h
                                                           mov ah,2
                                                           mov dl,10
  mov dl,13
  int 21h
                                                           int 21h
                                                           mov dl,13
    ;getting the 2nd input
                                                           int 21h
  mov ah,1
                                                           loop new ;again re initialize the code
  int 21h
                                                               exchanging the values;
```

;that means number1-number2=result

```
exchange:
  mov al,0
  add al,number1
  sub al,number2
  add al,48
  mov number2,al
  jmp print ;jumping to the print label
   ;ending the program
  mov ah,4ch
  int 21h
  main endp
end main
UVA 10055 C
#include<stdio.h>
int main()
  long long int a,b,c;
  while(scanf("%lld%lld",&a,&b)==2)
    if(a>b)
      c=a-b;
    else
      c=b-a;
    printf("%lld\n",c);
```

return 0;

}

```
UVA 10071 Assembly
                                                               ;else it will work
                                                              mov al,cl
org 100h
                                                              mul bl ;multiply al by bl and store it in al as
.model small
                                                           we know always result store in al
.stack 100h
                                                              mov dl,al ;moving the value from al to dl
.data
v db "initial velocity: $"
                                                              mov al,2 ; then store the value 2 in al
a db "initial acceleration: $"
                                                              mul dl ; and then multiply this al=2 by dl
result db "displacement be in twice of that time:
                                                           result value
$"
                                                              mov ah,0
.code
                                                              aam
                                                                       ;adjusting after multiply in al and ah
main proc
                                                           that means ax
    ;storing the data into the data segment
                                                                 ;making decimal value
  mov ax,@data
                                                              add ah,48
                                                              add al,48
  mov ds,ax
 first:
                                                                  ;moving ax to bx for our register work
    ;hint for velocity
                                                              mov bx,ax
  lea dx,v
  mov ah,9
                                                              ;hint for result
  int 21h
                                                             lea dx,result
      ;getting the first input
                                                             mov ah,9
  mov ah,1
                                                             int 21h
  int 21h
  sub al,48 ;make the ascii value to the decimal
                                                                 ;printing the bx value
  mov bl,al ;moving the value of al to the
                                                              mov dl,bh
variable a as a=al
                                                              mov ah,2
                                                              int 21h
    ;checking the input is zero or not
                                                              mov dl,bl
                                                              int 21h
  cmp bl,0
  je zero ;if zero then jump to the label zero
                                                              call printline
  call printline
                                                             jmp again
  ;hint for acceleration
  lea dx,a
  mov ah,9
                                                            zero:
  int 21h
                                                                ;printing zero as output
    ;getting the scnd input
                                                             mov dl,0
  mov ah,1
                                                             mov ah,2
  int 21h
                                                             int 21h
  sub al,48 ;make the ascii value to the decimal
                                                                ;printing new line
      ;checking the input is zero or not
                                                             mov dl,10
                                                             int 21h
  cmp al,0
  je zero ; if zero then jump to the label zero
                                                             mov dl.13
  mov cl,al
                                                             int 21h
  call printline
                                                                ;initiating the programme again
                                                            again:
```

```
jmp first

main endp
proc printline
mov ah,2
mov dl,10
int 21h
mov dl,13
int 21h
ret
printline endp
end main
```

UVA 10071 CPP

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
   int a,b;
   while(cin>>a>>b)
   {
      if(a==0 && b==0)
        cout<<0<<"\n";
      else
        cout<<2*a*b<<"\n";
   }
}</pre>
```

| UVA 10079 Assembly | add al,bl |
|--|---|
| | mov value,al |
| .model small | |
| .stack 100h | jmp input1 |
| .data | |
| prog db "Pizza Cutting.\$" | result2: |
| about db 10,13,"A negative number terminates | call line |
| the input.\$" | |
| value db? | mov result,1 ;taking the result value 1 |
| result db? | initially |
| count db? | mov cl,value ;moving input value in the cl |
| .code | registor |
| main proc | |
| ;fetching the data | mov count,1 ;making the count value 1 as |
| mov ax,@data | cutted pizza |
| mov ds,ax | P.223 |
| mov dojak | for: |
| ;loading the description of the program | ;changing the result value |
| lea dx,prog | mov bh,result |
| mov ah,9 | add bh,count |
| int 21h | mov result,bh |
| III(21II | illov result,bil |
| again, tronrogrammable label | inc count vinercomenting the count value |
| again: ;reprogrammable label | inc count ;increamenting the count value |
| lea dx,about | ada ayaa ayaa ah tha aa ayaa ay a laa ayaatiii it |
| mov ah,9 int 21h | ;decreament the counter value untill it |
| IIIL ZIII | will appear to zero dec cl |
| and time and time | |
| call line ;new line | cmp cl,0 |
| | je final ;if zero then we got the final result |
| mov value,0 ;making the storing variable | |
| value to zero | jmp for |
| | final: |
| input1: | ;making the two digit umber as 16 bit |
| taking input untill new line | register number in ascii |
| mov ah,1 | mov al,1 |
| int 21h | mov bl,result |
| mov bl,al | mul bl |
| sub bl,48 | mov ah,0 |
| | aam |
| comparing for negative number; | ;making the ascii to decimal |
| cmp al,45 | add al,48 |
| je exiting_input | add ah,48 |
| ;comparing for new line | mov bx,ax |
| cmp al,13 | |
| je result2 | ;printing the result |
| | mov ah,2 |
| ;else storing it as 10*value+bl | mov dl,bh |
| mov bh,value | int 21h |
| mov al,10 | mov dl,bl |
| mul bh | int 21h |

```
jmp again ; initialize the programme again
     ;when we will get - input then untill new line
appear we will take input
  exiting_input:
  mov ah,1
  int 21h
  cmp al,13
  je exit
  jmp exiting_input
      ;label for exiting the function
  exit:
  mov ah,4ch
  int 21h
  main endp
  ;function for a new line
proc line
  mov ah,2
  mov dl,10
  int 21h
  mov dl,13
  int 21h
  ret
  line endp
end main
```

UVA 10079 C

```
#include<stdio.h>
int main()
{
    long long int T,sum,i;
    while(scanf("%lld",&T)==1)
    {
        if(T<0)
            return 0;
        sum=1;
        for(i=0;i<=T;i++)
            sum+=i;
        printf("%lld\n",sum);
    }
    return 0;
}</pre>
```

```
UVA 10170 Assembly
                                                           je push_value ;|| n==d)
include "emu8086.inc"
                                                           inc i ;i++)
.model small
                                                           jmp for
.stack 100h
.data
                                                          push_value:
s dw?
                                                           mov ax,i
d dw?
                                                           cmp ax,0
                                                                     ;checking for i is 0 or not
idw?
                                                           je pop_value
n dw?
value dw?
                                                           xor dx,dx
count dw?
                                                           mov bx,10
.code
                                                           div bx
                                                                     ;sum/10
main proc
      ;fetching all data
                                                           push dx
                                                                     ;pushing last digit as reminder
  mov ax,@data
                                                           mov i,ax
  mov ds,ax
                                                           inc count
                                                                       ;value length increase
again:
  mov count,0
                ;count=0
                                                           jmp push_value
  mov value,0
                                                          pop_value:
  call input
                                                           mov ax, count
                ;scanf("%d",&s)!=
                                                                       ;checking for value length
  mov ax,value
                                                           cmp ax,0
  mov s,ax
                                                           je exit2
                                                           dec count
  printn
                                                           pop dx
  mov value,0
                                                           add dx,48
  call input
                                                           mov ah,2
                                                                      ;printing digit from stack
                 ;scanf("%d",&d)!=
  mov ax, value
                                                           int 21h
  mov d,ax
                                                           jmp pop_value
  printn
                                                          exit2:
                                                           printn
                                                           jmp again ;calliing the program again
  mov n,0
             ;n=0;
                                                          exit:
  mov ax,s
                                                           mov ah,4ch
  mov i,ax ;for(i=s;
                                                           int 21h
 for:
  mov ax,i
                                                           main endp
  add n,ax ;n+=i;
                                                         input proc
                                                           push ax
  mov ax,n
                                                           push bx
                                                           push cx ;saving all data if used
  cmp ax,d
  jle second_test ;if(n>d) ||
                                                           push dx
  jmp push_value
                                                         for loop:
 second_test:
                                                           mov ah,1
  mov ax,n
                                                           int 21h
                                                                     ;getting input
  cmp ax,d
                                                           cbw
```

```
cmp ax,13
  je exit3 ;checking whether it is new line
  cmp ax,32
  je exit3 ;checking whether it is space
  sub ax,48 ;making pure digit
  mov cx,ax ;cx=input
  mov ax, value
  mov bx,10 ;value=value*10
  mul bx
  add ax,cx ;recent result+input
  mov value,ax ;value=recent result
  jmp for_loop ;loop call
 exit3:
  pop dx
  рор сх
          ;restoring all registor value
  pop bx
  pop ax
  ret
  input endp
end main
```

UVA 10170 C

```
#include<stdio.h>
int main()
{
    long s,d,i,n;
    while(scanf("%ld%ld",&s,&d)==2)
    {
        n=0;
        for(i=s;; i++)
        {
            n+=i;
            if(n>d || n==d)
            {
                 printf("%ld\n",i);
                 break;
        }
        }
     }
     return 0;
}
```

| UVA 10300 Assembly | cmp j,ax |
|--|--|
| in alvida "amav9000 in all | jge exit_loop ;j <b;< td=""></b;<> |
| include "emu8086.inc" | |
| .model small | mov value,0 |
| .stack 100h | call input |
| .data | mov ax,value ;scanf("%d",&c); |
| ;int i,j,a,b,c,d,e; | mov c,ax |
| i dw ? | |
| j dw ? | printn |
| a dw ? | |
| b dw? | mov value,0 |
| c dw ? | call input |
| d dw? | mov ax,value ;scanf("%d",&d); |
| e dw? | mov d,ax |
| count dw? | |
| value dw ? | printn |
| sum dw ? | |
| .code | mov value,0 |
| main proc | call input |
| ;fetching all data | mov ax,value ;scanf("%d",&e); |
| mov ax,@data | mov e,ax |
| mov ds,ax | |
| mor asjan | printn |
| again: | printeri |
| mov count,0 | mov ax,c |
| mov count,o | mov bx,e |
| movvalue 0 | |
| mov value,0 | mul bx ;(c*e); |
| call input ;scanf("%d",&a) | |
| mov ax,value | add ax,sum |
| mov a,ax | (#) |
| | mov sum,ax ; sum=sum+(c*e); |
| printn | |
| | inc j ;j++) |
| mov i,0 ;for(i=0; | jmp loop_for |
| case_loop: | |
| mov ax,a | exit_loop: |
| cmp i,ax | mov ax,sum |
| jge exit_case_loop ;i <a;< td=""><td>cmp ax,0</td></a;<> | cmp ax,0 |
| | je print_num |
| mov sum,0 ;int sum=0; | |
| , , , | mov dx,0 |
| mov value,0 | mov ax,sum |
| call input | mov bx,10 ;for printing moving 1 by 1 digit in |
| mov ax,value ;scanf("%d",&b); | stack |
| mov b,ax | div bx |
| mov b,ax | GIV DX |
| nrintn | mov sum av |
| printn | mov sum,ax |
| movi0 ·for/i=0 | push dx |
| mov j,0 ;for(j=0 | incount |
| loop_for: | inc count |
| mov ax,b | mov cx,count ;counting stack length |

```
jmp exit_loop
 print_num:
  pop dx
  add dl,48
               ;poping 1 value and printing it
  mov ah,2
  int 21h
  loop print_num
  printn
  inc i
            ;i++)
  jmp case_loop
 exit_case_loop:
  jmp again
               ;restartting program
 exit:
  mov ah,4ch
  int 21h
  main endp
input proc
  push ax
  push bx
            ;taking all register in stack
  push cx
  push dx
 input_for:
  mov ah,1
  int 21h
            ;scanf
  cmp al,13
  je exit_for
  sub al,48
            ;converting byte to word
  cbw
  mov cx,ax
  mov ax, value
  mov bx,10
               ;saving value
  mul bx
  add ax,cx
  mov value,ax ;value=value*10 + scanf
  jmp input_for
 exit_for:
  pop ax
  pop bx
             ;restoring all register
  рор сх
```

```
pop dx
ret
input endp
end main
```

UVA 10300 C

```
#include<stdio.h>
int main()
{
    int i,j,a,b,c,d,e;
    while(scanf("%d",&a)==1)
    {
        for(i=0;i<a;i++)
        {
            int sum=0;
            scanf("%d",&b);
            for(j=0;j<b;j++)
            {
                 scanf("%d%d%d",&c,&d,&e);
                 sum=sum+(c*e);
            }
            printf("%d\n",sum);
        }
    }
    return 0;
}</pre>
```

| UVA 10302 Assembly | mov ax,b |
|--------------------------------|---|
| | cmp ax,0 ;checking for sum is 0 or not |
| include "emu8086.inc" | je pop_value |
| .model small | |
| .stack 100h | xor dx,dx |
| .data | mov bx,10 |
| a dw? | div bx ;sum/10 |
| b dw? | |
| c dw? | push dx ; pushing last digit as reminder |
| idw? | mov b,ax |
| value dw ? | , , |
| count dw? | inc count ;value length increase |
| .code | , |
| main proc | jmp push_value |
| ;fetching all data | Just breez-rener |
| mov ax,@data | pop_value: |
| mov ds,ax | mov ax,count |
| 1110 4 43,414 | cmp ax,0 ;checking for value length |
| again: | je exit2 |
| mov count,0 ;count=0 | dec count |
| mov count,o ,count=0 | dec count |
| mov value,0 | non dy |
| · | pop dx add dx,48 |
| call input | |
| mov ax,value ;scanf("%d",&a)!= | mov ah,2 ;printing digit from stack |
| mov a,ax | int 21h |
| mov b,ax | ima non volvo |
| | jmp pop_value |
| printn | exit2: |
| | printn |
| mov b,0 ;b=0; | jmp again ;calliing the program again |
| | exit: |
| mov i,1 ;for(i=1; | mov ah,4ch |
| for: | int 21h |
| mov ax,i | |
| cmp ax,a | main endp |
| jg push_value ;i<=a; | input proc |
| | push ax |
| mov ax,i | push bx |
| mov bx,i | push cx ;saving all data if used |
| mul bx | push dx |
| mul bx | |
| | for_loop: |
| mov c,ax ;c=i*i*i; | mov ah,1 |
| | int 21h ;getting input |
| mov ax,c | cbw |
| add b,ax ;b=b+c; | cmp ax,13 |
| | je exit3 ;checking whether it is new line |
| inc i ;i++) | |
| jmp for | cmp ax,32 |
| | je exit3 ;checking whether it is space |
| push_value: | |

```
sub ax,48 ;making pure digit
  mov cx,ax ;cx=input
  mov ax, value
  mov bx,10 ;value=value*10
  mul bx
  add ax,cx ;recent result+input
  mov value,ax ;value=recent result
  jmp for_loop ;loop call
 exit3:
  pop dx
  рор сх
          ;restoring all registor value
  pop bx
  pop ax
  ret
  input endp
end main
```

UVA 10302 C

```
#include<stdio.h>
int main()
{
    long int a,b,c,i;
    while(scanf("%Id",&a)!=EOF)
    {
        b=0;
        for(i=1;i<=a;i++)
        {
            c=i*i*i;
            b=b+c;
        }
        printf("%Id\n",b);
    }
    return 0;
}</pre>
```

UVA 10327 Assembly mov i,1 ;for(i=1; mov j,0 ;for(j=0; include "emu8086.inc" mov si,0 .model small for2: .stack 100h mov ax,a .data cmp i,ax a dw? jge print ;i<a; b dw 100 dup(0) for3: c dw? mov ax,a idw? dec ax idw? cmp j,ax d dw? jge increament ;j<a-1; value dw? count dw? mov ax,b[si] .code cmp ax,b[si+2] jle increament2 ;if(b[j]>b[j+1]) main proc ;fetching all data mov ax,@data mov bx,b[si] mov ds,ax mov ax,b[si+2] mov b[si+2],bx again: mov count,0 mov b[si],ax ;swap(b[j] and b[j+1]) ;count=0 mov value,0 inc d ;d++; call input ;scanf("%d",&a) increament2: mov ax, value inc j mov a,ax ;j++) inc si ;;index increasing printn inc si jmp for3 mov d,0 ;d=0; increament: mov i,0 ;for(i=0; inc i ;i++) mov si,0 mov si,0 for1: mov j,0; for(j=0;mov ax,a jmp for2 cmp i,ax print: jge before_for2 ;i<a; mov dx,d print "Minimum exchange operations:" mov value,0 cmp dx,0 call input jg push_value ;scanf("%d",&b[i]) mov ax, value add dl,48 mov b[si],ax mov ah,2 ;if result is zero int 21h printn jmp exit2 ;i++) inc i push_value: inc si mov ax,d inc si ;index increasing cmp ax,0 ;checking for d is 0 or not jmp for1 je pop_value

before_for2:

```
xor dx,dx
  mov bx,10
            ;d/10
  div bx
            ;pushing last digit as reminder
  push dx
  mov d,ax
  inc count
              ;value length increase
  jmp push_value
 pop_value:
  mov ax, count
              ;checking for value length
  cmp ax,0
  je exit2
  dec count
  pop dx
  add dx,48
  mov ah,2
             ;printing digit from stack
  int 21h
  jmp pop_value
 exit2:
  printn
  jmp again ; calliing the program again
 exit:
  mov ah,4ch
  int 21h
  main endp
input proc
  push ax
  push bx
  push cx ;saving all data if used
  push dx
for_loop:
  mov ah,1
  int 21h
            ;getting input
  cbw
  cmp ax,13
  je exit3 ;checking whether it is new line
  cmp ax,32
  je exit3 ;checking whether it is space
  sub ax,48 ;making pure digit
  mov cx,ax ;cx=input
  mov ax, value
  mov bx,10 ;value=value*10
  mul bx
```

```
add ax,cx ;recent result+input

mov value,ax ;value=recent result

jmp for_loop ;loop call

exit3:
   pop dx
   pop cx
   pop bx ;restoring all registor value
   pop ax
   ret
   input endp
end main
```

UVA 10327 C

```
#include<stdio.h>
int main()
{
  int a,b[1000],c,i,j,d;
  while(scanf("%d",&a)!=EOF)
  {
    d=0;
    for(i=0; i<a; i++)
       scanf("%d",&b[i]);
    for(i=1; i<a; i++)
       for(j=0; j<a-1; j++)
         if(b[j]>b[j+1])
         {
           c=b[j];
           b[j]=b[j+1];
           b[j+1]=c;
           d++;
         }
       }
    printf("Minimum exchange operations:
%d\n",d);
  }
  return 0;
}
```

| UVA 10346 Assembly | div bx |
|--|--|
| include "emu8086.inc" | mov z,ax ;z=a/b; |
| .model small | |
| .stack 100h | mov x,dx ;x=a%b; |
| .data | |
| a dw ? | add ax,dx |
| b dw? | |
| x dw ? z dw ? | mov a,ax ;a=z+x; |
| n dw ? | mov 2V 7 |
| sum dw ? | mov ax,z add sum,ax ;sum=sum+z; |
| value dw ? | auu suiii,ax ,suiii–suiii 2, |
| count dw? | jmp while |
| .code | jpc |
| main proc | print_sum: |
| ;fetching all data | mov ax,n |
| mov ax,@data | add sum,ax ;sum=sum+n |
| mov ds,ax | push_value: |
| | mov ax,sum |
| again: | cmp ax,0 ;checking for sum is 0 or not |
| mov count,0 ;count=0 | je pop_value |
| mov value,0 | xor dx,dx |
| call input | mov bx,10 |
| mov ax,value ;scanf("%d",&a)!= | div bx ;sum/10 |
| mov a,ax | |
| movvalue 0 | push dx ; pushing last digit as reminder |
| mov value,0 | mov sum,ax |
| call input mov ax,value ;scanf("%d",&b)!= | inc count ;value length increase |
| mov b,ax | me count ,value length merease |
| mov bjax | jmp push_value |
| printn | Jb basa.a.a |
| r · | pop_value: |
| cmp ax,1 | mov ax,count |
| jle exit ;if(b>1) then exit | cmp ax,0 ;checking for value length |
| | je exit2 |
| mov ax,a | dec count |
| mov n,ax ;n=a | |
| | pop dx |
| mov sum,0 ;sum=0 | add dx,48 |
| | mov ah,2 ;printing digit from stack |
| while: | int 21h |
| mov ax,a | <u> </u> |
| cmp ax,b | jmp pop_value |
| jl print_sum ;while(a>=b) | exit2: |
| | jmp again ;calliing the program again |
| xor dx,dx | exit: |
| mov ax,a | mov ah,4ch |
| mov bx,b ;a/b | int 21h |

```
main endp
input proc
  push ax
  push bx
  push cx ;saving all data if used
  push dx
for_loop:
  mov ah,1
  int 21h
            ;getting input
  cbw
  cmp ax,13
  je exit3 ;checking whether it is new line
  cmp ax,32
  je exit3 ;checking whether it is space
  sub ax,48 ;making pure digit
  mov cx,ax ;cx=input
  mov ax,value
  mov bx,10 ;value=value*10
  mul bx
  add ax,cx ;recent result+input
  mov value,ax ;value=recent result
  jmp for_loop ;loop call
 exit3:
  pop dx
  рор сх
          ;restoring all registor value
  pop bx
  pop ax
  ret
  input endp
end main
```

UVA 10346 C

```
#include<stdio.h>
int main()
{
    int a,b,x,z;
    while(scanf("%d%d",&a,&b)!=EOF && b>1)
    {
        int n=a, sum=0;
        while(a>=b){
        z=a/b;
        x=a%b;
        a=z+x;
        sum=sum+z;
        }
        printf("%d\n",sum+n);
    }
    return 0;
}
```

```
UVA 10469 Assembly
                                                             mov dl,bl
include "emu8086.inc"
                                                             add dl,48
                                                                           ;print low digit
.model small
                                                             mov ah,2
.stack 100h
                                                             int 21h
.data
a db?
                                                           restart:
bdb?
                                                             printn
.code
                                                             jmp again
                                                                          ;restart the program
main proc
                                                             main endp
                                                          input proc
 again:
                                                           input_loop:
  mov dl,0
                                                             mov ah,1
  call input
               ;taking first input
                                                             int 21h
                                                             cmp al,13
  mov a,dl
                                                             je return
  printn
                                                             sub al,48
                                                             mov bl,al
  mov dl,0
                                                             mov al,dl
  call input
                ;taking second input
                                                             mov bh,10
                                                             mul bh
  mov b,dl
                                                             add al,bl
                                                             mov dl,al
  printn
                                                            jmp input_loop
  mov bl,a
                                                           return:
  mov bh,b
                                                             ret
  xor bh,bl
               ;first ^ second
                                                             input endp
                                                          end main
  cmp bh,9
  jg greater
               ;if(output>9)
               ;print 2 digit
  mov dl,bh
                                                          UVA 10469 C
  add dl,48
                ;else
  mov ah,2
                  ;print 1 digit
                                                          #include <iostream>
  int 21h
  jmp restart
                                                          using namespace std;
 greater:
                                                          int main()
  mov al,bh
                                                          {
  xor ah,ah
                                                             int i, j;
                                                             while (cin >> i >> j)
  aaa
                                                               cout << (i ^ j) << '\n';
                                                          }
  mov bx,ax
```

mov dl,bh add dl,48

mov ah,2 int 21h

;print high digit

UVA 10499 Assembly

```
include "emu8086.inc"
.model small
.stack 100h
.data
.code
main proc
 input:
  mov ah,1
             ;scanf("%ld",&a)
  int 21h
  cmp al,45
  je finish
              ;if(a>0) then finish
  cmp al,1
  je zero_output ;if(a==1) then zero_output
  sub al,48
  mov bl,25
  mul bl
  mov ah,0
  mov bl,10
                ;25*a
  div bl
  mov bx,ax
  printn
  mov ah,2
  mov dl,bl
              ;as it is 2 digit value then prin high
value which is stored in bl register
  add dl,48
  int 21h
  mov dl,bh
  add dl,48
                 ;low value which is in bh
  int 21h
  printn "%"
  jmp input
 zero_output:
  printn "0%"
                   ;printf("0%%\n");
  jmp input
  finish:
  mov ah,1
              ;taking another any input after -
sign
  int 21h
  printn
  mov ah,4ch
                 ;exit
  int 21h
  main endp
end main
```

UVA 10499 C

```
#include<stdio.h>
int main()
{
    long int a;
    while(scanf("%ld",&a)==1 && a>0)
    {
        if(a==1)
            printf("0%%\n");
        else
            printf("%ld%%\n",25*a);
    }
    return 0;
}
```

```
UVA 10970 Assembly
                                                            mov dl,bl
include "emu8086.inc"
                                                            add dl,48
                                                                          ;print low digit
.model small
                                                            mov ah,2
.stack 100h
                                                            int 21h
.data
a db?
                                                          restart:
bdb?
                                                            printn
.code
                                                            jmp again
                                                                         ;restart the program
main proc
                                                            main endp
                                                         input proc
 again:
                                                          input_loop:
  mov dl,0
                                                            mov ah,1
  call input
               ;taking first input
                                                            int 21h
                                                            cmp al,13
  mov a,dl
                                                            je return
  printn
                                                            sub al,48
                                                            mov bl,al
  mov dl,0
                                                            mov al,dl
  call input
                ;taking second input
                                                            mov bh,10
                                                            mul bh
  mov b,dl
                                                            add al,bl
                                                            mov dl,al
  printn
                                                           jmp input_loop
  mov ah,0
                                                          return:
  mov al,a
                                                            ret
  mov bl,b
                                                            input endp
  mul bl
                                                         end main
  sub al,1
  cmp al,9
  jg greater
                                                         UVA 10970 C
  mov dl,al
  add dl,48
                ;else
                                                         #include<stdio.h>
  mov ah,2
                  ;print 1 digit
                                                         int main()
  int 21h
                                                         {
  jmp restart
                                                            int a,b,sum;
                                                            while(scanf("%d%d",&a,&b)!=EOF)
 greater:
                                                              if(1<=a<=300 && 1<=b<=300)
  xor ah,ah
                                                              {
  aaa
                                                                sum=n*m-1;
                                                                printf("%d\n",sum);
  mov bx,ax
                                                              }
  mov dl,bh
                                                           }
  add dl,48
               ;print high digit
                                                            return 0;
```

}

mov ah,2

int 21h

| include "emu8086.inc" | UVA 11150 Assembly | | jg greater_than_15 cmp al,9 | |
|---|--------------------|------------------------|---------------------------------------|---------------------|
| .data n db ? .code main proc again: mov dl,0 | | | • | |
| .data n db ? | .stack 100h | | mov dl,sum | |
| n db ? mov ah,2 print 1 digit sum db ? int 21h code pmp restart gagain: mov ah,0 mov dl,0 pig exit mov ah,0 mov n,dl mov dl,bl mov n,dl mov dl,bl mov sum,dl pig exit mov dl,bl mov n,dl mov dl,bl mov ab,0 mov dl,bl mov bh,1 print high digit mov ah,2 print high digit mov ah,2 print high digit mov ah,2 print low digit mov al,n mov al,a mov bh,3 print low digit mov al,a print fill wd n,a mov al,a printf("%d\n", sum); aaa printf(| .data | | | ;else |
| int 21h jmp restart jmp | | | | |
| jmp restart | | | · · | 71 |
| main proc greater_than_15: | | | | |
| again: | | | jp restare | |
| again: mov dl,0 | mam proc | | greater than | 15· |
| mov dl,0 (all input cmp dl,0 (pe sit) (scanf("%d", &n) (div bl mov bl,10 (div bl mov bx,ax) mov n,dl mov sum,dl mov sum,dl mov bl,10 (mov bh,10 (pe sit)) (mov dl,bl add dl,48 (mov ah,2 (mov ah, | again: | | | |
| call input mov bl,10 div bl mov bx,ax mov n,dl mov dl,bl mov sum,dl ;sum = 0; printn mov dl,bl while: mov dl,bh mov bh,n mov dl,bh cmp bh,3 ;while(n > 0) jl checking mov dl,bh mov al,n jmp restart mov al,n greater_than_9: mov bh,sum xor ah,ah ;printf("%d\n", sum); add ab,al ;n = n/3 + n%3; mov n,ah mov bx,ax add ab,al ;n = n/3 + n%3; mov n,ah mov dl,bh add dl,48 ;printf("%d\n", sum); aaa ;;print 2 digit mov bx,ax mov dl,bh add dl,48 ;print high digit mov bx,ax mov dl,bh add dl,48 ;print high digit mov bx,ax mov dl,bh add dl,48 ;print high digit mov ab,2 int 21h int 21h mov dl,bl add dl,48 ;print low digit int 21h mov dl,bl < | | :scanf("%d" &n) | · · | |
| cmp dl,0 ;compare al=0 div bl mov n,dl mov dl,bl add dl,48 ;print high digit while: mov dl,bh mov dl,bh cmp bh,3 ;while(n > 0) add dl,48 ;print low digit mov ah,0 mov ah,2 int 21h mov ah,0 jmp restart greater_than_9: mov bh,3 ;sum += n/3; greater_than_9: div bh mov al,sum xor ah,ah ;printf("%d\n", sum); add adh,al ;n = n/3 + n%3; mov bx,ax add ah,al ;n = n/3 + n%3; mov dl,bh mov bx,ax mov dl,bh add dl,48 ;print f("%d\n", sum); add adh,al ;n = n/3 + n%3; mov dl,bh add dl,48 ;print high digit jmp while mov dl,bh add dl,48 ;print high digit checking: mov al,2 int 21h checking: mov dl,bl add dl,48 ;print low digit mov al,2 int 21h add dl,48 ;print low digit mov al,2 int 21h add dl,48 ;print low | | ,scam(/od , em) | | |
| mov n,dl | • | :compare al=0 | | |
| mov n,dl mov sum,dl ;sum = 0; add dl,48 ;print high digit while: mov dl,bh mov dl,bh print low digit mov add dl,48 print low digit print low digit mov add dl,48 print low digit | | ,compare a:-0 | | |
| mov sum,dl ;sum = 0; add dl,48 mov ah,2 int 21h while: int 21h mov bh,n mov dl,bh add dl,48 mov ah,2 int 21h mov ah,0 mov ah,0 mov al,n mov bh,3 gives add bh,al mov bh,sum add bh,al mov sum,bh jmp restart greater_than_9: mov al,an printf("%d\n", sum); aaa ;;print 2 digit mov sum,bh mov bh,sum xor ah,ah printf("%d\n", sum); aaa ;;print 2 digit mov bh,ax mov dl,bh add dl,48 print high digit mov ah,a print 2 digit mov bh,ax mov dl,bh add dl,48 print high digit print aad dl,48 print high digit mov ah,a print aad dl,48 print high digit print aad dl,48 print high digit mov ah,a print aad sum inc sum print mov dl,bl add dl,48 print low digit add sum inc sum print restart: printn print print print print print print: mov al,sum exit: | je exit | | IIIOV DX,ax | |
| mov sum,dl ;sum = 0; add dl,48 ;print high digit mov ah,2 int 21h int 21h while: mov dl,bh add dl,48 ;print low digit mov ah,0 mov ah,2 int 21h mov ah,0 jmp restart greater_than_9: mov bh,3 ;sum += n/3; greater_than_9: div bh xor ah,ah ;printf("%d\n", sum); aaa ;;print 2 digit mov bh,sum aaa ;;print 2 digit mov sum,bh mov bx,ax add ah,al ;n = n/3 + n%3; mov dl,bh mov n,ah mov dl,bh add dl,48 ;print high digit jmp while mov ah,2 int 21h checking: mov ah,2 int 21h checking: mov dl,bl add dl,48 ;print low digit mov ah,2 int 21h int 21h add dl,48 ;print low digit mov ah,2 int 21h add_sum ;int 21h add_sum ;int 21h add_sum ;int 21h int 21h ;int 21h | mov n.dl | | mov dl.bl | |
| printn while: mov bh,n cmp bh,3 | | :sum = 0: | | nrint high digit |
| printn while: mov bh,n cmp bh,3 jl checking mov ah,0 mov ah,0 mov bh,3 div bh mov bh,sum add bh,al mov sum,bh add ah,al jmp restart mov n,ah checking: mov ah,2 je add_sum jmp print add_sum: inc sum print: mov ah,sum sint 21h mov dl,bh add dl,48 mov ah,2 int 21h mov dl,bh add dl,48 mov bx,ax mov dl,bh add dl,48 print low digit mov ah,2 int 21h mov dl,bh add dl,48 print high digit mov ah,2 int 21h restart: printn jmp again print: mov al,sum restart the program | mov sam,a | ,54 | | ,p8 a.8.c |
| while: mov bh,a | nrintn | | · · · · · · · · · · · · · · · · · · · | |
| mov bh,n mov dl,bh cmp bh,3 ;while(n > 0) jl checking mov ah,2 mov ah,0 jmp restart mov ah,3 jmp restart greater_than_9: mov al,sum wor ah,ah ;printf("%d\n", sum); add bh,al aaa mov sum,bh mov bx,ax add ah,al ;n = n/3 + n%3; mov n,ah mov dl,bh jmp while mov ah,2 checking: mov ah,2 mov al,sum mov dl,bl add dl,48 ;print low digit mov al,2 int 21h add dl,48 ;print low digit mov al,2 int 21h add dl,48 ;print low digit mov al,2 int 21h add dl,48 ;print low digit mov al,2 int 21h add sum: int 21h inc sum ;n++; print: pmp again ;restart the program print: mov al,sum | • | | 111(2111 | |
| cmp bh,3 ;while(n > 0) jl checking mov ah,0 mov al,n mov bh,3 ;sum += n/3; div bh mov bh,sum add bh,al mov sum,bh add ah,al ;n = n/3 + n%3; mov n,ah checking: mov ah,n cmp ah,2 je add_sum jmp print add_sum: inc sum ;n++; mov al,sum xor ah,ah sprintf("%d\n", sum); aaa ;;print 2 digit mov bx,ax mov dl,bh add dl,48 ;print high digit mov ah,2 int 21h restart: printn jmp again ;restart the program exit: | | | mov dl hh | |
| mov ah,2 int 21h mov ah,2 int 21h mov ah,2 int 21h jmp restart greater_than_9: mov al,sum xor ah,ah ;printf("%d\n", sum); aaa ;;print 2 digit mov bx,ax add ah,al ;n = n/3 + n%3; mov n,ah add ah,al ;m = n/3 + n%3; mov n,ah checking: mov ah,n cmp ah,2 je add_sum ;if(n==2) then add_sum jmp print add_sum: inc sum ;n++; print: mov al,sum mov ah,2 int 21h mov ah,2 int 21h restart: print jmp again ;restart the program print: mov al,sum exit: | | y | | enrint low digit |
| int 21h mov ah,0 mov al,n mov bh,3 ;sum += n/3; div bh mov bh,sum add bh,al mov sum,bh add ah,al ;n = n/3 + n%3; mov n,ah checking: mov ah,n cmp ah,2 je add_sum ;if(n==2) then add_sum jmp print add_sum: inc sum ;n++; mov al,sum int 21h jmp restart greater_than_9: mov al,sum xor ah,ah ;printf("%d\n", sum); aaa ;;print 2 digit mov bx,ax mov dl,bh add dl,48 ;print high digit mov ah,2 int 21h restart: printn jmp again ;restart the program print: mov al,sum exit: | | ,wille(11 > 0) | | ,print low digit |
| mov ah,0 mov al,n mov bh,3 div bh mov bh,sum add bh,al mov sum,bh add ah,al jmp restart greater_than_9: mov al,sum xor ah,ah mov bx,ax add ah,al jmp while checking: mov ah,n cmp ah,2 je add_sum jmp print add_sum: inc sum jmp restart greater_than_9: mov al,sum xor ah,ah jprintf("%d\n", sum); aaa jprint 2 digit mov dl,bh add dl,48 jprint high digit mov ah,2 int 21h restart: printn jmp again jrestart the program exit: | Ji Checking | | | |
| mov al,n mov bh,3 ;sum += n/3; div bh mov bh,sum add bh,al mov sum,bh add ah,al ;n = n/3 + n%3; mov n,ah checking: mov ah,n cmp ah,2 je add_sum jmp print add_sum: inc sum ;n++; mov al,sum xor ah,ah ;printf("%d\n", sum); aaa ;;print 2 digit mov bx,ax mov dl,bh add dl,48 ;print high digit mov ah,2 int 21h mov dl,bl add dl,48 ;print low digit mov ah,2 int 21h restart: printn jmp again ;restart the program print: mov al,sum exit: | mov ah 0 | | IIIL ZIII | |
| mov bh,3 ;sum += n/3; div bh mov bh,sum add bh,al mov sum,bh add ah,al ;n = n/3 + n%3; mov n,ah checking: mov ah,n cmp ah,2 je add_sum jmp print add_sum: inc sum ;n++; mov al,sum xor ah,ah ;printf("%d\n", sum); aaa ;;print 2 digit mov bx,ax mov dl,bh add dl,48 ;print high digit mov ah,2 int 21h mov dl,bl add dl,48 ;print low digit mov ah,2 int 21h restart: printn jmp again ;restart the program print: mov al,sum exit: | | | imp restart | |
| div bh mov bh,sum add bh,al mov sum,bh add ah,al ;n = n/3 + n%3; mov n,ah checking: mov ah,n cmp ah,2 je add_sum jmp print add_sum: inc sum print: mov al,sum xor ah,ah xor ah,ah yprintf("%d\n", sum); aaa ;;print 2 digit mov bx,ax mov dl,bh add dl,48 print high digit mov ah,2 int 21h add dl,48 print low digit mov ah,2 int 21h restart: printn jmp again print: mov al,sum xor ah,ah print 2 digit mov bx,ax mov dl,bh add dl,48 print low digit mov ah,2 int 21h restart: printn jmp again prints exit: | | isum 1= n/2; | | 0. |
| mov bh,sum add bh,al mov sum,bh add ah,al ;n = n/3 + n%3; mov n,ah checking: mov ah,n cmp ah,2 je add_sum jmp print add_sum: inc sum ;n++; mov al,sum xor ah,ah ;printf("%d\n", sum); aaa ;;print 2 digit mov bx,ax mov dl,bh add dl,48 ;print high digit mov ah,2 int 21h add dl,48 ;print low digit mov ah,2 int 21h restart: printn jmp again ;restart the program print: mov al,sum xor ah,ah ;printf("%d\n", sum); aaa ;;print 2 digit mov bx,ax mov dl,bh add dl,48 ;print low digit restart: printn jmp again ;restart the program exit: | | ;Sum += n/3; | | _9: |
| add bh,al mov sum,bh add ah,al ;n = n/3 + n%3; mov n,ah add ah,al ;n = n/3 + n%3; mov dl,bh add dl,48 ;print high digit mov ah,2 int 21h checking: mov ah,n cmp ah,2 je add_sum ;if(n==2) then add_sum jmp print add_sum: inc sum ;n++; printn jmp again ;restart the program print: mov al,sum exit: | | | I | |
| mov sum,bh add ah,al ;n = n/3 + n%3; mov n,ah mov dl,bh add dl,48 ;print high digit mov ah,2 int 21h checking: mov ah,n cmp ah,2 je add_sum ;if(n==2) then add_sum jmp print add_sum: inc sum ;n++; printn jmp again ;restart the program print: mov al,sum mov bx,ax mov dl,bh add dl,48 ;print low digit restart: printn jmp again ;restart the program exit: | | | | |
| add ah,al ;n = n/3 + n%3; mov n,ah impowed,bh add dl,48 ;print high digit mov ah,2 int 21h checking: mov ah,n cmp ah,2 je add_sum jmp print add_sum: inc sum print: mov al,sum mov bx,ax mov dl,bh add dl,48 ;print low digit mov ah,2 int 21h restart: printn jmp again ;restart the program exit: | • | | aaa | ;;print 2 digit |
| add ah,al ;n = n/3 + n%3; mov n,ah mov dl,bh add dl,48 ;print high digit mov ah,2 int 21h checking: mov ah,n cmp ah,2 je add_sum jmp print add_sum: inc sum ;n++; printn jmp again ;restart the program print: mov al,sum mov dl,bl add dl,48 ;print low digit mov ah,2 int 21h restart: printn jmp again ;restart the program exit: | mov sum,bn | | | |
| mov n,ah mov dl,bh add dl,48 int 21h checking: mov ah,n cmp ah,2 je add_sum jimp print add_sum: inc sum print: mov al,sum mov dl,bl add dl,48 int 21h mov dl,bl add dl,48 int 21h restart: printn jmp again irestart the program exit: | امامامام | /2 | mov bx,ax | |
| add dl,48 ;print high digit mov ah,2 int 21h checking: mov ah,n cmp ah,2 je add_sum ;if(n==2) then add_sum jmp print add_sum: inc sum ;n++; printn jmp again ;restart the program print: mov al,sum ;print high digit mov ah,2 int 21h add dl,48 ;print low digit mov ah,2 int 21h restart: printn jmp again ;restart the program exit: | | ;n = n/3 + n%3; | بادا الدينية | |
| jmp while mov ah,2 int 21h checking: mov ah,n mov dl,bl add dl,48 ;print low digit mov ah,2 jmp print mov ah,2 int 21h add_sum: restart: inc sum ;n++; printn jmp again ;restart the program print: mov al,sum exit: | mov n,an | | | Leader de . e |
| int 21h checking: mov ah,n cmp ah,2 je add_sum jmp print add_sum: inc sum jn++; inc sum print: mov al,sum int 21h mov dl,bl add dl,48 print low digit mov ah,2 int 21h restart: printn jmp again print: exit: | | | | print nigh digit |
| checking: mov ah,n cmp ah,2 je add_sum ;if(n==2) then add_sum jmp print add_sum: inc sum ;n++; printn jmp again ;restart the program print: mov al,sum mov dl,bl add dl,48 ;print low digit mov ah,2 int 21h restart: printn jmp again ;restart the program exit: | Jmp while | | | |
| mov ah,n cmp ah,2 je add_sum ;if(n==2) then add_sum jmp print add_sum: inc sum ;n++; printn jmp again ;restart the program print: mov al,sum mov dl,bl add dl,48 ;print low digit mov ah,2 int 21h restart: printn jmp again ;restart the program exit: | | | int 21h | |
| cmp ah,2add dl,48;print low digitje add_summov ah,2jmp printint 21hadd_sum:restart:inc sum;n++;printnprint:jmp again;restart the programprint:exit: | | | | |
| je add_sum ;if(n==2) then add_sum mov ah,2 int 21h add_sum: restart: printn jmp again ;restart the program print: mov al,sum exit: | | | I | |
| jmp print add_sum: inc sum ;n++; printn jmp again ;restart the program print: mov al,sum int 21h restart: printn jmp again ;restart the program exit: | | | | ;print low digit |
| add_sum: inc sum ;n++; printn jmp again ;restart the program print: mov al,sum exit: | - | ;if(n==2) then add_sum | | |
| inc sum ;n++; printn jmp again ;restart the program print: mov al,sum exit: | jmp print | | int 21h | |
| inc sum ;n++; printn jmp again ;restart the program print: mov al,sum exit: | add arres | | rootot- | |
| jmp again ;restart the program print: mov al,sum exit: | _ | | | |
| print: mov al,sum exit: | inc sum | ;п++; | · · | |
| mov al,sum exit: | | | Jmp again | restart the program |
| · | • | | - 4 | |
| cmp al,15 mov ah,4ch | | | | |
| | cmp al,15 | | mov an,4ch | |

```
int 21h
  main endp
input proc
input_loop:
  mov ah,1
  int 21h
  cmp al,13
  je return
  sub al,48
                ;taking multiple value digit
  mov bl,al
  mov al,dl
  mov bh,10
  mul bh
  add al,bl
  mov dl,al
  jmp input_loop
 return:
  ret
  input endp
end main
```

UVA 11150 C

| UVA 11172 Assembly | mov ah,2 | | |
|---|--|--|--|
| | mov dl,10 | | |
| org 100h | int 21h | | |
| .model small | mov dl,13 | | |
| .stack 100h | int 21h | | |
| .data | | | |
| a db "enter two digit number or 1 digit along 0 | ;hint for scnd two digit number | | |
| in the first case",10,13,"such as 12(two) or | lea dx,c | | |
| 01(one)\$" | mov ah,9 | | |
| b db 10,13,"for exit input two 0(zero)\$" | int 21h | | |
| c db "input number 2: \$" | | | |
| d db 10 | mov al,a ;moving the frst digit to al | | |
| e db 10,13,"input number 1: \$" | mul d ;multiplicate the value of al by d=10 | | |
| .code | mov ah,0 ;vacant the ah | | |
| main proc | aam ;adjusting after multiplication the ax | | |
| ;loading data into the data segment | add al,b ;then adding the al with b which will | | |
| mov ax,@data | compatible in ax | | |
| mov ds,ax | compatible in ax | | |
| mov us,ax | getting the ascii value; | | |
| unvinting the hint massage | | | |
| ;printing the hint message | add ah,48 | | |
| lea dx,a | add al,48 | | |
| mov ah,9 | | | |
| int 21h | mov bx,ax ;moving the ax to bx | | |
| again: | ;checking if the input is zero | | |
| ;the label for re starting the code | cmp bh,48 | | |
| ;hint for exiting the code | je zero1 | | |
| lea dx,b | | | |
| mov ah,9 | zero1: | | |
| int 21h | cmp bl,48 | | |
| | je exit ;if ax==00 then exit | | |
| ;input message for input 1 | ;else continue | | |
| lea dx,e | , | | |
| mov ah,9 | ;input frst digit of 2digits number | | |
| int 21h | mov ah,1 | | |
| | int 21h | | |
| ;input frst digit of 2digits number | sub al,48 | | |
| mov ah,1 | mov a,al ;moving the frst value to the a | | |
| int 21h | variable | | |
| sub al,48 | Variable | | |
| mov a,al ;moving the frst value to the a | ;input the scnd number of 2digits number | | |
| variable | mov ah,1 | | |
| variable | int 21h | | |
| ;input the scnd number of 2digits number | sub al,48 | | |
| mov ah,1 | mov b,al ;moving the scnd value to the b | | |
| int 21h | variable | | |
| | variable | | |
| sub al,48 | unqueling | | |
| mov b,al ;moving the scnd value to the b | ;newline | | |
| variable | mov ah,2 | | |
| an accelling | mov dl,10 | | |
| ;newline | int 21h | | |

```
mov dl,13
  int 21h
  mov al,a ;moving the frst digit to al
           ;multiplicate the value of al by d=10
  mul d
  mov ah,0 ;vacant the ah
  aam
           ;adjusting after multiplication the ax
  add al,b ;then adding the al with b which will
compatible in ax
      ;getting the ascii value
  add ah,48
  add al,48
  mov cx,ax
              ;moving the ax to cx
    ;comparing either 2 high byte is equal
  cmp bh,ch
  je equal
     ;comparing either frst high is big than 2nd
  cmp bh,ch
  jg greater
  jmp less
  equal:
  ;comparing either 2 low byte is equal
  cmp bl,cl
  je equal2
    ;comparing either last low is big than 2nd
  cmp bl,cl
  jg greater
  jmp less
  equal2:
  ;printing the equal message
  mov dl,"="
  mov ah,2
  int 21h
  imp newline
  greater:
  ;printing the greater message
  mov dl,">"
  mov ah,2
  int 21h
  jmp newline
```

```
;printing the lesser message
 mov dl,"<"
 mov ah,2
 int 21h
 imp newline
   ;newline
 newline:
 mov ah,2
 mov dl,10
 int 21h
 mov dl,13
 int 21h
 jmp again
 exit:
 exiting the program;
 mov ah,4ch
 int 21h
 main endp
end main
```

UVA 11172 C

```
#include<stdio.h>
int main()
{
    int a,b,c,i;
    while(scanf("%d",&a)==1)
    {
        for(i=0; i<a; i++)
        {
            scanf("%d%d",&b,&c);
            if(b>c)
                printf(">\n");
            else if(b<c)
                printf("<\n");
            else
                printf("=\n");
        }
        return 0;
}</pre>
```

| UVA 11479 Assembly | mov bl,a |
|----------------------------------|-------------------------------------|
| | add bl,c |
| include "emu8086.inc" | cmp bl,b ;if((a+c)<=b)) |
| .model small | jl invalid ;jump invalid |
| .stack 100h | |
| .data | mov bl,a |
| a db ? | cmp bl,0 ;if(a==0) |
| b db ? | jl invalid ;jump invalid |
| c db ? | |
| case db 49 | mov bl,a |
| .code | cmp bl,0 ;if(b==0) |
| main proc | jl invalid ;jump invalid |
| ;fetching data from data segment | |
| mov ax,@data | mov bl,a |
| mov ds,ax | cmp bl,0 ;if(c==0) |
| | jl invalid ;jump invalid |
| getting the test case | |
| mov ah,1 | mov bl,a |
| int 21h | cmp bl,b ;if(a==b) |
| sub al,48 | je andtrue ;jump and true |
| mov cl,al ;count=testcase | |
| mov ch,0 | mov bl,b |
| printn ;\n | cmp bl,c |
| | je Isosceles |
| all: | |
| mov ah,1 | mov bl,a |
| int 21h | cmp bl,c |
| sub al,48 ;taking input a | je Isosceles |
| mov a,al | |
| print " " | ;printf("Case %ld: Scalene\n",i); |
| | print "Case " |
| int 21h | mov ah,2 |
| sub al,48 ;taking input b | mov dl,case |
| mov b,al | int 21h |
| print " " | printn ": Scalene" |
| | |
| int 21h | jmp again_start |
| sub al,48 ;taking input c | |
| mov c,al | andtrue: |
| printn | mov bl,b |
| | cmp bl,c ;if(a==b) && (b==c) |
| mov bl,a | je Equilateral ;jump equilateral |
| add bl,b $;if((a+b) \le c))$ | |
| cmp bl,c | |
| jl invalid ;jump invalid | ;printf("Case %ld: Isosceles\n",i); |
| | Isosceles: |
| mov bl,b | print "Case " |
| add bl,c | mov ah,2 |
| cmp bl,a ;if((c+b)<=a)) | mov dl,case |
| jl invalid ;jump invalid | int 21h |
| | |

```
printn ": Isosceles"
  jmp again_start
    ;printf("Case %ld: Equilateral\n",i);
 Equilateral:
  print "Case "
  mov ah,2
  mov dl,case
  int 21h
  printn ": Equilateral"
  jmp again_start
   ;printf("Case %ld: Invalid\n",i);
 Invalid:
  print "Case "
  mov ah,2
  mov dl,case
  int 21h
  printn ": Invalid"
 again_start:
  mov al, case
  inc al
           ;case++
  mov case, al
  loop all
            ;testcase++
  ;exiting the program
 exit:
  mov ah,4ch
  int 21h
  main endp
end main
```

UVA 11479 C

```
#include<stdio.h>
int main()
  long int t,a,b,c,i;
  while(scanf("%ld",&t)==1)
    i=1;
    while(i<=t)
       scanf("%ld%ld%ld",&a,&b,&c);
       if((a+b) \le c \mid | (b+c) \le a \mid | (c+a) \le b)
         printf("Case %ld: Invalid\n",i);
       else if(a<=0 || b<=0 || c<=0)
         printf("Case %ld: Invalid\n",i);
       else if(a==b \&\& b==c)
         printf("Case %ld: Equilateral\n",i);
       else if(a==b || b==c || c==a)
         printf("Case %Id: Isosceles\n",i);
       else
         printf("Case %ld: Scalene\n",i);
       i++;
    }
  return 0;
```

```
UVA 11498 Assembly
                                                              mov first,dl
include 'emu8086.inc'
                                                              printn
.model small
                                                                  ;taking the input and moving it to the scnd
.stack 100h
                                                           variable
.data
                                                             xor dl,dl
  ;variable and data section
                                                              call input
testcase db?
                                                              mov scnd,dl
first db?
scnd db?
                                                              mov cl, testcase ; taking the testcase to the
test1 db?
                                                           counter register for making the loop length
test2 db?
                                                              printn
    ;string for declaring the result
divisa db "divisa",10,13,"$"
                                                                  ;now taking the two number for testcase
ne db "NE",10,13,"$"
                                                           length to check
se db "SE",10,13,"$"
                                                            testing input:
no db "NO",10,13,"$"
                                                                  ;checking the counter value whether zero or
so db "SO",10,13,"$"
                                                           not
.code
                                                              cmp cl,0
  ;code section
                                                             je last ;if zero then terminate to the restarting
main proc
  ;main procedure start
                                                              dec cl
  mov ax,@data
  mov ds,ax
                                                                 ;taking the first input for divisia checking
                                                             xor dl.dl
        1
                                                              call input
    NO | NE
                                                              mov test1,dl
  -----divisa-----
                                                              printn
    SO | SE
                                                                  ;taking the scnd input
                                                             xor dl,dl
                                                              call input
                                                              mov test2,dl
  ;label for restarting the program
                                                                  ;moving to the another register for
 again:
  xor dl,dl ;making the dl registor value initially
                                                           checking
                                                              mov bh,test1
  call input ;calling the input fuction for taking
                                                              mov bl,test2
input
  mov testcase,dl
                                                                   ;comparing the first input to the first fixed
                                                           divisia value
        ;comparing the input test case if zero
                                                              cmp bh,first
                                                              jg first_greater ;if greater then go to label first
  cmp testcase,0
  je exit ;if zero then the program will terminate
                                                           greater
                                                             jl first_less
                                                                              ;else go to the first_less label
  printn ;printing a new line
                                                                    ;if all above are wrong then given and
      ;again taking the input and moving it to the
                                                           exist must be equal
first variable
                                                            equal:
  xor dl,dl
                                                              printn
  call input
                                                                  ;if equal then priting the divisia string
```

```
lea dx, divisa
                                                                   ;if bh<first and bl<scnd then it will print NE
  mov ah,9
                                                               lea dx,so
  int 21h
                                                               mov ah,9
  loop testing_input
                                                               int 21h
                                                              loop testing_input
 first greater:
  cmp bl,scnd
                                                             last:
  jg firstgreater_scndgreater ;if bl>scnd then
                                                                 ; when loop will end then the last label will be
jump to the label firstgreater_scndgreater
                                                            called
  jl firstgreater_scndless
                                ;else to the label
                                                              jmp again
firstgreater_scndless
                                                                   ;the exiting label
  imp equal
               ;if all above are wrong then it
                                                             exit:
must be equal
                                                               mov ah,4ch
                                                               int 21h
 first less:
                                                               main endp
  cmp bl,scnd
  jg firstless scndgreater ;if bl>scnd then jump to
                                                               ;procedure for taking input
the label firstless_scndgreater
                                                            input proc
  jl firstless_scndless
                          ;;else to the label
                                                              for:
firstless scndless
                                                               mov ah,1
                                                              int 21h
  jmp equal
                ;if all above are wrong then it
                                                               cmp al,13 ;if input is new line the it will
must be equal
                                                            refused to take input
                                                              je exit for
 firstgreater scndgreater:
                                                               sub al,48 ;taking the ascii value
                                                                            ;bh=10
  printn
                                                               mov bh,10
      ;if bh>first and bl>scnd then it will print NE
                                                               mov bl,al
                                                                            ;bl=input value al
  lea dx,ne
                                                               mov al,dl ;stored value dl to the al
  mov ah,9
                                                               mul bh
                                                                            ;al=al*bh
  int 21h
                                                               add al,bl
                                                                            ;al=al+bl
  loop testing_input
                                                               mov dl,al
                                                                            ;dl=al final
                                                              jmp for ;again taking input
 firstgreater_scndless:
  printn
                                                                ;when exit for called then it will return the
      ;if bh>first and bl<scnd then it will print NE
                                                            ascii value store in dl
  lea dx,se
                                                             exit_for:
  mov ah,9
                                                              ret
  int 21h
                                                               input endp
  jmp last
                                                            end main
 firstless scndgreater:
  printn
      ;if bh<first and bl>scnd then it will print NE
  lea dx,no
  mov ah,9
  int 21h
  loop testing_input
 firstless_scndless:
  printn
```

UVA 11498 C

```
#include<stdio.h>
int main()
{
  int t,m,n,x,y;
  while(scanf("%d",&t)==1 && t!=0)
    scanf("%d%d",&m,&n);
    while(t-->0)
      scanf("%d%d",&x,&y);
      if(x==m || y==n)
        printf("divisa\n");
      else if(x>m && y>n)
        printf("NE\n");
      else if(x>m && y<n)
        printf("SE\n");
      else if(x<m && y>n)
        printf("NO\n");
      else if(x<m && y<n)
        printf("SO\n");
    }
  }
  return 0;
```

| UVA 11877 Assembly | | jmp while | |
|---|------------------------------|---|--|
| include "emu80 .model small .stack 100h .data n db ? sum db ? .code | 086.inc" | print: mov al,sum cmp al,15 jg greater_tha cmp al,9 jg greater_tha | |
| main proc | | mov dl,sum | .alaa |
| again: mov dl,0 call input | ;scanf("%d", &n) | add dl,48 mov ah,2 int 21h jmp restart | ;else ;print 1 digit |
| cmp dl,0 | ;compare a!=0 | | |
| je exit mov n,dl mov sum,0 | ;sum = 0; | greater_than_ mov ah,0 mov al,sum mov bl,10 div bl | 15: |
| printn | | mov bx,ax | |
| while: | | | |
| mov bh,n cmp bh,0 jle print | ;while(n > 0) | mov dl,bl add dl,48 mov ah,2 int 21h | ;print high digit |
| mov ah,0 mov al,n mov bh,3 div bh mov bh,sum | ;sum += n/3; | mov dl,bh add dl,48 mov ah,2 int 21h | ;print low digit |
| add bh,al mov sum,bh add ah,al mov n,ah | ;n = n/3 + n%3; | jmp restart greater_than_ mov al,sum xor ah,ah aaa | 9: ;printf("%d\n", sum); ;;print 2 digit |
| cmp ah,2 je checking cmp ah,1 | ;if(a==2 | mov bx,ax | |
| je checking jmp while | ; a==1) then jump checking | mov dl,bh add dl,48 mov ah,2 int 21h | ;print high digit |
| jmp print | ;if(a==2) then add_n | mov dl,bl add dl,48 mov ah,2 int 21h | ;print low digit |
| add_n: inc n ;n+ | ++; | restart: | |

```
printn
  jmp again
               ;restart the program
 exit:
  mov ah,4ch
  int 21h
  main endp
input proc
 input_loop:
  mov ah,1
  int 21h
  cmp al,13
  je return
  sub al,48
                ;taking multiple value digit
  mov bl,al
  mov al,dl
  mov bh,10
  mul bh
  add al,bl
  mov dl,al
  jmp input_loop
 return:
  ret
  input endp
end main
```

UVA 11877 C

```
#include<stdio.h>
int main()
  int a,b,c,sum;
  while(scanf("%d",&a)==1 && a!=0)
    sum=0;
    while(a>0){
    b=a/3;
    c=a%3;
    a=b+c;
    sum=sum+b;
    if(a==2 || a==1)
      if(a==2)
         a+=1;
      else
        break;
    }
    }
    printf("%d\n",sum);
  return 0;
```

UVAS 12646 Assembly

```
include 'emu8086.inc'
                        ;importing a header file
.model small
.stack 100h
.data
;variable declaration
adb?
bdb?
cdb?
.code
main proc
    ;fetching data into the data segment
  mov ax,@data
  mov ds,ax
    ;restarting the program
      ;outer while(true)
 again:
            ;for(int i=1)
  mov bl,1
    ;for loop start
 input_for:
  cmp bl,3
              ;checking i<=3?
  ig operation
  mov ah,1
               ;scanf
  int 21h
  sub al,48
  cmp bl,1
  je invoke_a ;if(bl==1) then jumping tothe label
invoke_a
  cmp bl,2
  je invoke b ;if(bl==2) then jumping tothe label
invoke_b
  mov c,al
               ;c=al
  inc bl
  jmp input for ;again for loop start with bl++
 invoke a:
  mov a,al
                :a=al
  inc bl
  jmp input_for
                   ;again for loop start with bl++
 invoke b:
  mov b,al
                 ;b=al
  inc bl
  jmp input_for
                  ;again for loop start with bl++
```

```
;after loop end ooperation will start
operation:
  printn
           ;new line
  mov bl,a
  cmp bl,0
  je a_zero ;if a==0
             ;else a==1
 jmp a_one
a_zero:
  mov bl,b
  cmp bl,0
  je a_zero_b_zero ;if a==0 and b==0
 jmp a_zero_b_one ;else a==0 and b==1
 a one:
  mov bl,b
  cmp bl,0
 je a_one_b_zero ;if a==1 and b==0
 jmp a_one_b_one ;else a==1 and b==1
a_zero_b_one:
  mov bl.c
  cmp bl,0
  je a_zero_b_one_c_zero
                           ;if a==0 and b==1
and c==0
 jmp a zero b one c one
                            ;else a==0 and
b==1 and c==1
a_zero_b_zero:
  mov bl,c
  cmp bl,1
  je a_zero_b_zero_c_one
                           ;if a==0 and b==0
and c==1
  jmp last
              ;else jump last
a_one_b_zero:
  mov bl,c
  cmp bl,0
  je a_one_b_zero_c_zero ;if a==1 and b==0
and c==0
  jmp a_one_b_zero_c_one ;else a==1 and
b==0 and c==1
a_one_b_one:
```

mov bl,c

```
cmp bl,0
  je a_one_b_one_c_zero ;if a==1 and b==1
and c==0
  jmp last
                  ;else jump last
 a_zero_b_zero_c_one:
  printn "C"
                  ;printf "C"
  jmp repro
 a_zero_b_one_c_zero:
                  ;printf "B"
  printn "B"
  jmp repro
 a_zero_b_one_c_one:
  printn "A"
                  ;printf "A"
  jmp repro
 a_one_b_zero_c_zero:
  printn "A"
                  ;printf "A"
  jmp repro
 a_one_b_zero_c_one:
  printn "B"
                  ;printf "B"
 jmp repro
 a_one_b_one_c_zero:
                  ;printf "C"
  printn "C"
 jmp repro
last:
                  ;printf "*"
  printn "*"
   ;starting again that means while loop continue
 repro:
  jmp again
  main endp
end main
```

UVA 12646 C

```
#include<stdio.h>
int main()
{
  int a,b,c;
  while(scanf("%d%d%d",&a,&b,&c)==3 && (a==0
|| a==1 || b==0 || b==1 || c==0 || c==1))
      if((a==0 && b==0 && c==1) || (a==1 &&
b==1 \&\& c==0)
        printf("C\n");
      else if((a==0 && b==1 && c==0) || (a==1
&& b==0 && c==1))
        printf("B\n");
      else if((a==1 && b==0 && c==0) || (a==0
&& b==1 && c==1))
        printf("A\n");
      else
        printf("*\n");
     return 0;
}
```

| UVA 12700 Assembly | cmp i,bl |
|---|---|
| | je exit |
| include 'emu8086.inc' | inc i ;i++ |
| .model small | |
| .stack 100h | ;initializing all variable 0 |
| .data | mov b,0 |
| ;making all the variable | mov w,0 |
| testout db? | mov t,0 |
| b db? | mov a,0 |
| w db? | ,- |
| t db? | ;scanf |
| a db ? | xor dl,dl |
| i db ? | call input |
| j db ? | printn ;new line |
| testin db ? | mov testin,dl ;testin=scanf(dl) |
| casestring db "Case \$" | mov j,0 ;int j=0 |
| abandoned db ": ABANDONED\$" | ;for loop 2 |
| whitewash db ": WHITEWASH\$" | testcasein: |
| • | |
| banglawash db ": BANGLAWASH\$" draw db ": DRAW \$" | ;checking j<=testin |
| • | mov bl,testin |
| bangladesh db ": BANGLADESH \$" | cmp j,bl |
| www db ": WWW \$" | je next_phase ;after loop ending ->next phase |
| case db? | inc j ;j++ |
| about db "These letters will be either `B' or `W' or | |
| `T' or `A'.",10,13,"\$" | getting another input |
| about_2 db "here B=Bangladesh W=WWW , T=Tie | mov ah,1 |
| and A=Abandoned",10,13,"\$" | int 21h |
| .code | ;dl=got input |
| main proc | mov dl,al |
| ;fetching the data variable | |
| mov ax,@data | ;checking the input either =A |
| mov ds,ax | cmp dl,65 |
| | je aplus |
| ;about string output | checking the input either =B; |
| lea dx,about | cmp dl,66 |
| mov ah,9 | je bplus |
| int 21h | checking the input either =T; |
| lea dx,about_2 | cmp dl,84 |
| int 21h | je tplus |
| | checking the input either =W; |
| ;taking the testcase input | cmp dl,87 |
| xor dl,dl | je wplus |
| call input ;scanf called(function) | |
| mov testout,dl ;testout=scanf(dl) | ;if all are false then jumping testcasein for |
| mov i,0 ;int i=0 | taking again input |
| mov case,49 ;int case=1(49 in ascii) | jmp testcasein |
| ;initial outer for loop | , , |
| testcaseout: | ;label for a++ |
| printn ;new line | aplus: |
| ;i<=testout | inc a |
| mov bl,testout | jmp testcasein |
| • | , , , , , , , , , , , , , , , , , , , |

```
;label for b++
bplus:
                                                          others:
inc b
                                                          mov bl,b
                                                          cmp bl,w
jmp testcasein
  ;label for t++
                                                          je draw_final
                                                                          ;if(b==w)
tplus:
                                                                       ;then draw final label
inc t
                                                          cmp bl,w
                                                          jg bangladesh_final ;else if(b>w)
imp testcasein
  ;label for w++
                                                                          ;then bangladesh final label
wplus:
                                                          jmp www_final
                                                                            ;else
                                                                      ;then www final
inc w
                                                                ;printing the whitewash string
imp testcasein
                                                          whitewash_final:
next_phase:
                                                          printn
mov bl,b
                                                          call casing
cmp bl,0
                                                          lea dx, whitewash
je b zero ;checking if b==0
                                                          mov ah,9
          ;else checking if w==0
                                                          int 21h
mov bl,w
                                                          jmp last
cmp w,0
                                                               ;printing the banglawash string
                                                          banglawash_final:
je w_zero
                                                          printn
jmp others
               ;or to the others label
                                                          call casing
                                                          lea dx,banglawash
   ;if(b==0)
                                                          mov ah,9
b zero:
                                                          int 21h
mov bl,w
                                                          jmp last
cmp bl,0
                                                               ;printing the abandoned string
                    ;if b==0 and w==0
                                                          abandoned final:
je b_zero_w_zero
                                                          printn
mov bl,t
                                                          call casing
                                                          lea dx,abandoned
cmp bl,0
je whitewash_final ;if(b==0 and t==0)
                                                          mov ah,9
               ;then whitewash label
                                                          int 21h
                                                          imp last
jmp others
                                                              ;printing the draw with score string
                                                          draw_final:
w zero:
                                                          printn
mov bl,t
                                                          call casing
cmp bl,0
                                                          lea dx,draw
je banglawash_final
                       ;if(w==0 and t==0)
                                                          mov ah,9
                 ;then banglawash label
                                                          int 21h
jmp others
                                                          mov ah,2
                                                          mov bl,b
b_zero_w_zero:
                                                          add bl,48
mov bl,testin
                                                          mov dl,bl
cmp bl,a
                                                          int 21h
                                                          mov dl,"-"
je abandoned final
                       ;if(b==0 and w==0 and
a==testin)
                                                          int 21h
                 ;then abandoned label
                                                          mov bl,t
```

```
add bl,48
                                                           main endp
                                                             ;procedure for printing the case string with
mov dl,bl
int 21h
                                                           number
jmp last
                                                           casing proc
      ;printing the bangladesh_final with score
                                                           lea dx, casestring
string
                                                           mov ah,9
bangladesh final:
                                                           int 21h
printn
                                                           mov dl,case
call casing
                                                           mov ah,2
lea dx,bangladesh
                                                           int 21h
mov ah,9
int 21h
                                                           inc case
                                                           ret
mov ah,2
                                                           casing endp
                                                           ;procedure for taking input
mov bl,b
add bl,48
                                                           input proc
mov dl,bl
                                                           for:
int 21h
                                                           mov ah,1
mov dl,"-"
                                                           int 21h
int 21h
                                                           cmp al,13 ;if input is new line the it will refused
mov bl,w
                                                           to take input
add bl,48
                                                           je exit_for
mov dl,bl
                                                           sub al,48 ;taking the ascii value
int 21h
                                                                       ;bh=10
                                                           mov bh,10
imp last
                                                           mov bl.al
                                                                        ;bl=input value al
   ;printing the www final with score string
                                                           mov al,dl ;stored value dl to the al
                                                                        ;al=al*bh
www_final:
                                                           mul bh
printn
                                                           add al,bl
                                                                        ;al=al+bl
call casing
                                                           mov dl,al
                                                                        ;dl=al final
lea dx,www
                                                           jmp for ;again taking input
mov ah,9
int 21h
                                                           ; when exit for called then it will return the ascii
                                                           value store in dl
mov ah,2
                                                           exit_for:
mov bl,w
                                                           ret ;returning the input value
add bl,48
                                                           input endp
mov dl,bl
                                                           end main
int 21h
mov dl,"-"
int 21h
mov bl,b
add bl,48
mov dl.bl
int 21h
last:
jmp testcaseout
;exit
exit:
mov ah,4ch
int 21h
```

```
UVA 12700 C
```

```
#include<stdio.h>
int main()
  int ti,i,m,b,w,a,t,cas=1;
  char n;
  scanf("%d",&ti);
  while(ti-->0)
    b=0;
    w=0;
    t=0;
    a=0;
    scanf("%d",&m);
    for(i=0; i<=m; i++)
      scanf("%c",&n);
      switch (n)
      {
      case 'B':
        b++;
        break;
      case 'W':
        w++;
        break;
      case 'T':
        t++;
        break;
      case 'A':
        a++;
        break;
      }
    }
      if(b==0 && w==0 && a==m)
        printf("Case %d: ABANDONED\n",cas++);
      else if(w==0 && t==0)
        printf("Case %d:
BANGLAWASH\n",cas++);
      else if(b==0 \&\& t==0)
        printf("Case %d: WHITEWASH\n",cas++);
      else if(b>w)
        printf("Case %d: BANGLADESH %d -
%d\n",cas++,b,w);
      else if(b<w)
        printf("Case %d: WWW %d -
%d\n",cas++,w,b);
      else if(b==w)
        printf("Case %d: DRAW %d
%d\n",cas++,b,t);
  }
```

return 0;
}

UVA 12917 Assembly cmp bl,cl jg hunter ;if 1+2>3 then jump to the hunter include 'emu8086.inc' label .model small ;else .stack 100h ;printing the props wining string .data lea dx,props hint db "'Props win!' if the props survive, mov ah,9 otherwise print `Hunters win!'.\$" int 21h hunters db "Hunters win!\$" printn props db "Props win!\$" jmp again ;jumping to the again label adb? bdb? hunter: cdb? ;printing the props wining string .code lea dx, hunters main proc mov ah,9 ;fetching the data int 21h mov ax,@data printn mov ds,ax jmp again main endp ;printing the hint data ;procedure for taking input lea dx,hint input proc mov ah,9 for: int 21h mov ah,1 printn int 21h cmp al,13 ; if input is new line the it will ;restarting the program again refused to take input je exit for again: ;getting the first input sub al,48 ;taking the ascii value xor dl,dl mov bh,10 ;bh=10 call input mov bl,al ;bl=input value al mov al,dl ;stored value dl to the al mov a,dl mul bh ;al=al*bh printn ;getting the second input add al,bl ;al=al+bl xor dl.dl mov dl.al :dl=al final call input jmp for ;again taking input mov b,dl printn ;when exit for called then it will return the ;getting the third input ascii value store in dl xor dl,dl exit for: call input ret ;returning the input value mov c,dl input endp printn end main ;moving the 3 data to the registor mov bl,a mov bh,b mov cl,c

;adding the first 2 data

;comparing the adding the data to the 3rd

add bl,bh

data

UVA 12917 C

```
#include<stdio.h>
int main()
{
    int p,h,o;
    while(scanf("%d%d%d",&p,&h,&o)==3)
    {
        if(p+h>o)
            printf("Hunters win!\n");
        else
            printf("Props win!\n");
    }
    return 0;
}
```

```
UVA 12952 Assembly
                                                            mov dl,10
                                                            int 21h
org 100h
                                                            mov dl,13
.model small
                                                            int 21h
.stack 100h
                                                               ;output message
.data
                                                            lea dx,win
about db "a program to determine the value of
                                                            mov ah,9
the third card",10,13,"that maximizes the
                                                            int 21h
probability of that ",10,13," player winning the
                                                                ;checking what is greater than other
game.$"
                                                            cmp bl,bh
frst db 10,13,"Enter the frst number A (1<A<13):
                                                            jge greater; if bl greater than and equal with bh
",10,13,"$"
                                                          then jump greater label
scnd db 10,13,"Enter the second number B
                                                            jmp lesser
                                                                            ;else lesser label
(1<B<13): ",10,13,"$"
win db "Winning Card: ",10,13,"$"
                                                           greater:
                                                               ;printing greater or equal as bl
.code
                                                            mov dl,bl
main proc
                                                            mov ah,2
                                                            int 21h
  ;loading data to the data segment
  mov ax,@data
                                                                ;procedure start again for next test case
  mov ds,ax
                                                            jmp again_start
   ;loading the about
                                                           lesser:
  lea dx,about
                                                              ;printing leasser as bh
  mov ah,9
                                                            mov dl,bh
  int 21h
                                                            mov ah,2
                                                            int 21h
again start:
      ;setting message for first input
                                                               ;procedure start again for next test case
  lea dx,frst
                                                            jmp again_start
  mov ah,9
                                                               main endp
  int 21h
                                                          end main
                                                          UVA 12952 C
    ;getting the first input
  mov ah,1
  int 21h
                                                          #include<stdio.h>
                                                          int main()
  mov bh,al
                                                          {
     ;setting message for second input
                                                            int a,b;
                                                            while(scanf("%d%d",&a,&b)==2)
  lea dx,scnd
  mov ah,9
  int 21h
                                                               if(a==b)
                                                                 printf("%d\n",a);
    ;getting the second input
                                                               else if(a>b)
  mov ah,1
                                                                 printf("%d\n",a);
  int 21h
                                                              else
                                                                 printf("%d\n",b);
  mov bl,al
    ;new line
                                                            return 0;
```

mov ah,2

```
mov bl,0
UVA 12992 Assembly
org 100h
                                                           input:
.model small
.stack 100h
                                                             ;showing the message for getting the nth
.data
                                                         value case
;defining the hint mesagge
                                                             lea dx,nth value
about db "what's the minimal number of bottles
                                                              mov ah,9
needed",10,13,"if he want to bring N types of
                                                              int 21h
medicine.$"
test_case db 10,13,"give the test case
                                                              ;getting the input which nth value we need
number:",10,13,"$"
                                                              mov ah,1
nth_value db 10,13,"give the desired n'th value:
                                                              int 21h
",10,13,"$"
                                                              mov cl,al
minimal_number db 10,13,"the minimal number
of bottles Huatuo needed: ",10,13,"$"
                                                              mov sum,1 ;making the value of sum initially
case print db 10,13,"Case %$"
                                                         1
case db?
                                                           getting_sum:
sum db?
                                                              dec cl ; for calculating nth value decreasing it
idb?
                                                             cmp cl,48 ;comparing either the nth is zero
.code
                                                         or not
main proc
                                                                        ;if true then print the sum
                                                             je print
    ;load the data value to the data segment
                                                                  ;checking either the value of sum is
    mov ax,@data
                                                         greater than 9 or not
    mov ds,ax
                                                              mov ch,sum
                                                              cmp ch,9
    ;load the about of the programme
                                                             jg greater nine ;if true then only increament
                                                         the lower part of bx as bl
    lea dx,about
    mov ah,9
    int 21h
                                                                 ;else increasing the sum value
                                                                 ;increamenting value of sum by 2 in every
        ;loading the program again
                                                         cases
  again programme:
                                                              inc ch
    ;showing the message for getting the input
                                                              inc ch
test case
                                                              mov sum,ch
    lea dx,test case
    mov ah,9
                                                                ;storing the sum value in ax registor not
    int 21h
                                                         bothering about 2 digits
                                                             mov al,0
    ;getting the input
                                                              add al,sum
    mov ah,1
                                                              mov ah,0
    int 21h
                                                              aaa
    mov case,al
                                                              add ah,48 ;making ascii the higher part
    mov i,49 ;moving case value as 1
                                                              add al,48 ;making ascii the higher part
         ;making the value of bx zero
                                                              mov bx,ax ;storing the ax value to the bx
    mov bh,0
```

```
;again checking the loop
    jmp getting_sum
getting_sum
  greater_nine: ;if greater nine then
increament only lower part
    inc bl
             ;once
    inc bl
              ;twice
    jmp getting_sum
                       ;again checking the loop
getting_sum
 print:
      ;printing the case message
    mov ah,9
    lea dx,case_print
    int 21h
       ;printing the case number
    mov ah,2
    mov dl,i
    int 21h
    mov dl,":"
    int 21h
    cmp sum,9
    jle print_less
       ;priniting the sum value from bx registor
    mov ah,2
    mov dl,bh
    int 21h
    mov dl,bl
    int 21h
  inc_case:
       ;increamenting the case value
    mov bh,i
    inc bh
    mov i,bh
       ;if case value is lesser than input test case
then again get input
    cmp bh,case
    ile input
          ;else programme restarting
    jmp again_programme
  print_less:
      ;printing the value less or equal 9
    add sum,48
    mov ah,2
    mov dl,sum
```

```
int 21h

jmp inc_case ;jumping to the increment case

;programme exit exit:
 mov ah,4ch int 21h

main endp end main
```

UVA 12992 C

```
#include<stdio.h>
int main()
{
  int t,sum,n,i,cas=1;
  scanf("%d",&t);
  while(t-->0)
    scanf("%d",&n);
    i=1;
    sum=1;
    while(i!=n)
      sum+=2;
      i++;
    printf("Case #%d: %d\n",cas++,sum);
  }
  return 0;
}
```

int 21h **UVA 13012 Assembly** org 100h input: .model small ;taking the input .stack 100h mov ah,1 .data int 21h about db ":it is about priniting how many number",10,13," are containing in below 5 extra ;space checking as ascii value of space is 32 answer:(for 1 digit 1 only)",10,13,"\$" cmp al,32 correct_ans db 10,13,"enter the correct je input ;if true then again taking the input answer:(with 1 digit)",10,13,"\$" student ans db "give the 5 ans in 1 digit with ;comparing either the given ans either correct space or not:",10,13,"\$" or not how_correct db 10,13,"the correct ans are: cmp bh,al ",10,13,"\$" je increament ;if true then it will increament .code the flag and loop length main proc inc cl ;else it will only increase the loop length mov ax,@data mov ds,ax cmp cl,52 ;checking either the length is end or not lea dx,about jg print ;if true then jmp to print how many mov ah,9 are right int 21h jmp input ;jumping to the input label to take next input program start: ;correct ans input hint increament: lea dx,correct_ans ; label for increamenting the flag and loop mov ah,9 length int 21h inc bl inc cl ;taking the first correct answer input mov ah,1 cmp cl,52 ;checking either the length is end int 21h or not mov bh,al jg print ;if true then jmp to print how many are right ;printing a new line jmp input ;else again taking input mov ah,2 mov dl,10 print: int 21h ;Student correct ans number means flag mov dl,13 lea dx,how_correct int 21h mov ah,9 int 21h mov bl,48 ;moving the flag value how many are ;prtinting the flag value bl correct value 0 as 48 ascii mov ah,2 mov dl,bl mov cl,48 ;moving the loop length value 0 as 48 ascii int 21h ;Student 5 ans input hint imp program start ; again starting the lea dx,correct_ans programme to take input

mov ah,9

```
exit:
    ;exiting the programme
mov ah,4ch
int 21h

main endp
end main
```

UVA 13012 C

```
#include<stdio.h>
int main()
{
    int sum,i,a,b;
    while(scanf("%d",&a)==1)
    {
        sum=0;
        for(i=0;i<5;i++)
        {
            scanf("%d",&b);
            if(a==b)
            sum++;
        }
        printf("%d\n",sum);
    }
    return 0;
}</pre>
```

```
UVA 13018 Assembly
                                                           cmp bl,n
                                                                         ;if(m==n)
                                                           je print_m
include "emu8086.inc"
.model small
                                                           jmp looping_print_n
.stack 100h
.data
                                                          print_m:
m db?
                                                             add bl,1
ndb?
                                                             cmp bl,9
idb?
                                                             jg greater1
tdb?
                                                             mov dl,bl
.code
                                                                         ;printf("%d\n",m+1); for one
main proc
                                                             add dl,48
      ;fetching all data
                                                         digit
  mov ax,@data
                                                             mov ah,2
  mov ds,ax
                                                             int 21h
  mov t,1
                                                             imp after print m
             ;t=1;
 again:
  xor dl,dl
                                                            greater1:
  call input
             ;scanf("%d",&n)
                                                             mov al,bl
  mov m,dl
                                                             mov ah,0
                                                             mov bl,10
  printn
                                                             div bl
  xor dl,dl
                                                             mov cx,ax
             ;scanf("%d",&n)
  call input
                                                             mov dl,cl
                                                                           ;;printf("%d\n",m+1); for two
  mov n,dl
                                                         digit
  mov bl,t
              ;if(t==0)
                                                             add dl,48
  cmp bl,0
                                                             mov ah,2
  je one_line_print ;then print one line
                                                             int 21h
  jmp next_step
                                                             mov dl,ch
 one_line_print:
                                                             add dl.48
  printn
               ;printf("\n");
                                                             mov ah,2
                                                             int 21h
 next_step:
  mov bl,m
                                                          after_print_m:
  cmp bl,n
                                                           printn
                                                                       ;print new line
  jg just_exchange ;if(m>n)
                                                           jmp making_t_zero
  jmp just_exchange_skip ;else skip exchanging
just_exchange:
                                                          looping_print_n:
                                                           mov bl,m
  mov bl,m
                                                           add bl,1
                                                           mov i,bl
  mov bh,n
                                                                       ;for(i=m+1;)
  mov m,bh
                ;m=n
                                                           for_loop:
  mov n,bl
                ;n=m
                                                             mov bh,n
                                                             add bh,1
just_exchange_skip:
                                                             cmp i,bh
                                                                           ;i<=n+1
  mov bl,m
                                                             jg making_t_zero
```

```
cmp bl,9
    jg greater2
    mov dl,i
    add dl,48
                ;printf("%d\n",i); for one digit
    mov ah,2
    int 21h
    jmp loop_again
  greater2:
    mov al,i
    mov ah,0
    mov bl,10
    div bl
    mov cx,ax
    mov dl,cl
    add dl,48
                 ;;printf("%d\n",i); for 2 digit
    mov ah,2
    int 21h
    mov dl,ch
    add dl,48
    mov ah,2
    int 21h
   loop_again:
    printn
    inc i
             ;;i++
    jmp for_loop
 making_t_zero:
  mov t,0
              ;t=0;
  jmp again
  main endp
input proc
 input_loop:
  mov ah,1
  int 21h
  cmp al,13 ;taking input and checking either
new line
  je return
  sub al,48
  mov bl,al
  mov al,dl
  mov bh,10
               ;taking multi digit by adding with
10*before
```

```
mul bh ;+current
add al,bl
mov dl,al
jmp input_loop

return:
ret
input endp
end main
```

UVA 13018 C

```
#include<stdio.h>
int main()
  int m,n,temp,i,t=1;
  while(scanf("%d%d",&m,&n)==2)
  {
    if(t==0)
      printf("\n");
    if(m>n){
      temp=m;
      m=n;
      n=temp;
    }
    if(m==n)
      printf("%d\n",m+1);
    for(i=m+1;i<=n+1;i++)
      printf("%d\n",i);
    t=0;
  }
  return 0;
```

UVA 13025 Assembly

```
org 100h
.model small
.stack 100h
.data
date db "May 29, 2013 Wednesday",10,13,"$"
      ;defining the desired date
.code
main proc
  ;fetching the data
  mov ax,@data
  mov ds,ax
  ;printing the result
  lea dx,date
  mov ah,9
  int 21h
   ;exiting the program
  mov ah,4ch
  int 21h
  main endp
end main
```

UVA 13025 C

```
#include<stdio.h>
int main()
{
  printf("May 29, 2013 Wednesday\n");
  return 0;
}
```