

COAL LAB#12 TASKS

Name: Saman Khan

ID: 19K-0354

Section: H

TASK#01:

.asm:-

.686

.MODEL FLAT, C

.STACK 2048

.data

clear PROTO

GCD PROTO, temp1:DWORD, temp2:DWORD

.code

clear PROC

xor eax, eax

xor ecx, ecx

ret

clear endp

GCD PROC, temp1:DWORD, temp2:DWORD

mov eax,temp1

mov ecx,temp2

cmp eax,ecx

jae L1

mov eax,temp1

mov ecx,temp2

mov temp1,ecx

mov temp2,eax

mov temp1,eax

mov temp2,ecx

L1:

mov edx,0

div cx

mov eax,temp2

mov temp1,eax

mov temp2,edx

mov eax,temp1

mov ecx,temp2

cmp ecx,0

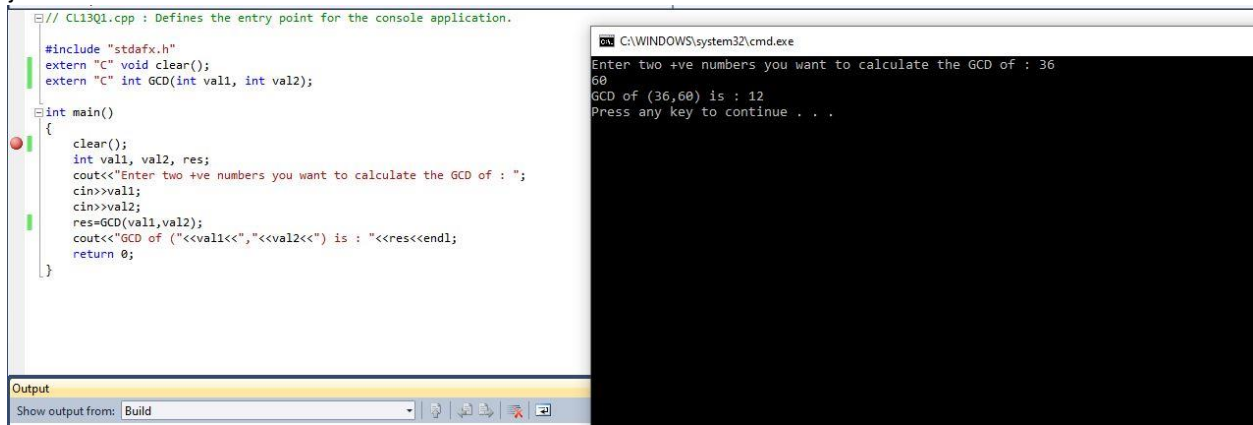
ja L1

ret

```
GCD endp  
end
```

.cpp:-

```
#include "stdafx.h"  
extern "C" void clear();  
extern "C" int GCD(int val1, int val2);  
  
int main()  
{  
    clear();  
    int val1, val2, res;  
    cout<<"Enter two +ve numbers you want to calculate the GCD of : ";  
    cin>>val1;  
    cin>>val2;  
    res=GCD(val1,val2);  
    cout<<"GCD of ("<<val1<<","<<val2<<) is : "<<res<<endl;  
    return 0;  
}
```



TASK#02:

.asm:-

```
.686  
.MODEL FLAT, C  
.STACK 2048  
  
.data  
clear PROTO  
Add_Three PROTO, temp1:SDWORD, temp2:SDWORD, temp3:SDWORD  
  
.code  
clear PROC  
xor eax, eax  
xor ecx, ecx  
ret
```

clear endp

Add_Three PROC, temp1:SDWORD, temp2:SDWORD, temp3:SDWORD

mov eax, temp1

mov ecx, temp2

add eax,ecx

add eax, temp3

ret

Add_Three endp

end

.cpp:-

#include "stdafx.h"

extern "C" void clear();

extern "C" int Add_Three(int val1, int val2, int val3);

int main()

{

clear();

int val1, val2, val3, res;

cout<<"Enter three numbers : ";

cin>>val1;

cin>>val2;

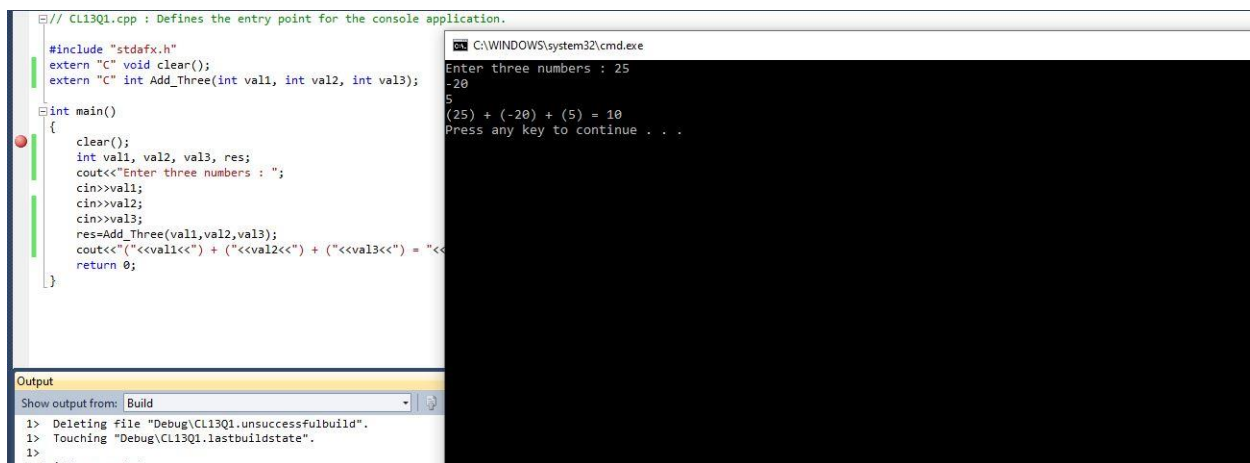
cin>>val3;

res=Add_Three(val1,val2,val3);

cout<<"("<<val1<<") + ("<<val2<<") + ("<<val3<<") = "<<res<<endl;

return 0;

}



The screenshot displays a C++ development environment. On the left, the source code for 'CL13Q1.cpp' is visible, which includes the necessary headers, declares the 'clear' and 'Add_Three' functions, and implements the 'main' function. The 'main' function prompts the user for three integers, calculates their sum using the 'Add_Three' function, and displays the result. On the right, the console window shows the program's execution. The user has entered the values 25, -20, and 5. The program correctly calculates the sum as 10 and displays the full expression: '(25) + (-20) + (5) = 10'. The bottom status bar indicates that the build was successful.

TASK#03:

.asm:-

.686

```
.MODEL FLAT, C
.STACK 2048
```

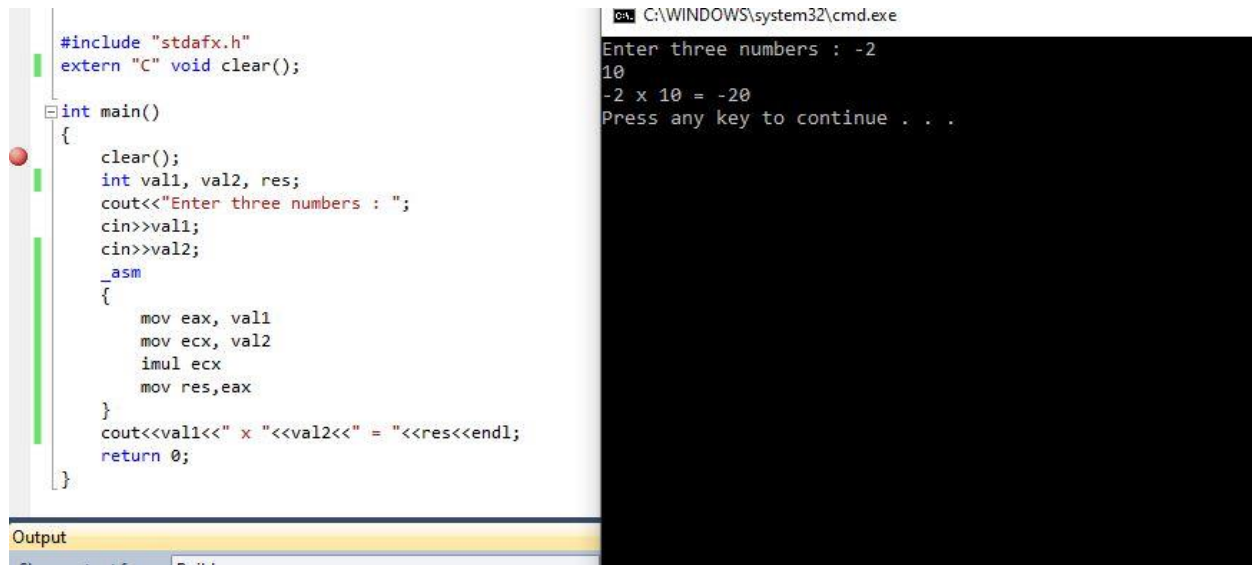
```
.data
clear PROTO
```

```
.code
clear PROC
xor eax, eax
xor ecx, ecx
ret
clear endp
end
```

.cpp:-

```
#include "stdafx.h"
extern "C" void clear();
```

```
int main()
{
    clear();
    int val1, val2, res;
    cout<<"Enter three numbers : ";
    cin>>val1;
    cin>>val2;
    _asm
    {
        mov eax, val1
        mov ecx, val2
        imul ecx
        mov res,eax
    }
    cout<<val1<<" x "<<val2<<" = "<<res<<endl;
    return 0;
}
```



```
#include "stdafx.h"
extern "C" void clear();

int main()
{
    clear();
    int val1, val2, res;
    cout<<"Enter three numbers : ";
    cin>>val1;
    cin>>val2;
    _asm
    {
        mov eax, val1
        mov ecx, val2
        imul ecx
        mov res, eax
    }
    cout<<val1<<" x "<<val2<<" = "<<res<<endl;
    return 0;
}
```

```
C:\WINDOWS\system32\cmd.exe
Enter three numbers : -2
10
-2 x 10 = -20
Press any key to continue . . .
```

TASK#04:

.asm:-

.686

.MODEL FLAT, C

.STACK 2048

.data

min SDWORD ?

max SDWORD ?

clear PROTO

MinMax PROTO, X: PTR DWORD

.code

clear PROC

xor eax, eax

xor ebx, ebx

ret

clear endp

MinMax PROC, X: PTR DWORD

mov ecx, 10

dec ecx

mov esi, X

mov eax, [esi]

L1:

add esi, 4

mov ebx, [esi]

cmp ebx, eax

jl next

```

jmp skip
next:
mov eax, ebx
skip:
LOOP L1

```

```

mov min, eax

```

```

mov ecx, 10
dec ecx
mov esi, X
mov eax, [esi]
L2:
add esi, 4
mov ebx, [esi]
cmp ebx, eax
jg next2
jmp skip2
next2:
mov eax, ebx
skip2:
LOOP L2

```

```

mov max, eax
mov eax, min
mov ebx, max
ret
MinMax endp
end

```

.cpp:-

```

#include "stdafx.h"
extern "C" void clear();
extern "C" void MinMax(int *arr);

int main()
{
    clear();
    int min, max, arr[10];
    cout<<"Enter 10 numbers in the array : "<<endl;
    for(int i=0; i<10; i++)
    {
        cin>>arr[i];
    }
    MinMax(arr);
    _asm
    {

```

```

        mov min, eax
        mov max, ebx
    }
    cout<<"Min = "<<min<<endl<<" Max = "<<max<<endl;
    return 0;
}

```

The screenshot shows a C++ IDE with two main panes. The left pane displays the source code for a program that finds the minimum and maximum values in an array. The right pane shows the output of the program, which prompts the user to enter 10 numbers and then displays the calculated minimum and maximum values.

Source Code (Left Pane):

```

// CL13Q1.cpp : Defines the entry point for the console application.
#include "stdafx.h"
extern "C" void clear();
extern "C" void MinMax(int *arr);

int main()
{
    clear();
    int min, max, arr[10];
    cout<<"Enter 10 numbers in the array : "<<endl;
    for(int i=0; i<10; i++)
    {
        cin>>arr[i];
    }
    MinMax(arr);
    _asm
    {
        mov min, eax
        mov max, ebx
    }
    cout<<"Min = "<<min<<endl<<" Max = "<<max<<endl;
    return 0;
}

```

Output (Right Pane):

```

C:\WINDOWS\system32\cmd.exe
Enter 10 numbers in the array :
-30
5
20
60
4
-100
200
90
1
2
Min = -100
Max = 200
Press any key to continue . . .

```