

1. Program Code **Provide your code as text, not as a screenshot/image.**

Results **Please provide your output with a screenshot/image.**

INCLUDE Irvine32.inc

```
.386
.model flat,stdcall
.stack 4096
ExitProcess proto,dwExitCode:dword
.data
numberOfCustomers DWORD 0
numberOfDrinks DWORD 0
numberOfSandwiches DWORD 0
sizeOfSandwiches DWORD 0
costOfWater DWORD 1
costOfSoda DWORD 2
costOfTenInSandwich DWORD 3
costOfTwelveInSandwich DWORD 5
totalBill DWORD 0
typeOfDrink BYTE '0'
textString BYTE "-----7-11 Convenience Store-----", 0
textString2 BYTE "Drinks: ", 0
textString3 BYTE "Soda(S) - $2", 0
textString4 BYTE "Water(W) - $1", 0
textString5 BYTE "Sandwiches: ", 0
textString6 BYTE "10 inches - $3", 0
textString7 BYTE "12 inches - $5", 0
prompt BYTE "Enter the number of customers: ", 0
prompt2 BYTE "How many drinks? ", 0
prompt3 BYTE "What kind of drink (S = Soda, W = Water - If It's Not S Then You Get Water B/C We Only  
Have Two Types Of Drinks)? ", 0
prompt4 BYTE "How many sandwiches? ", 0
prompt5 BYTE "What size of sandwich (10/12 - MUST choose ONE)? ", 0
answer BYTE "Your total bill = $", 0
.code
main proc
    mov edx, OFFSET textString
    call WriteString
    call CrLf
    mov edx, OFFSET textString2
    call WriteString
    call CrLf
    mov edx, OFFSET textString3
    call WriteString
    call CrLf
    mov edx, OFFSET textString4
    call WriteString
    call CrLf
    mov edx, OFFSET textString5
```

```
call WriteString
call CrLf
mov edx, OFFSET textString6
call WriteString
call CrLf
mov edx, OFFSET textString7
call WriteString
call CrLf
call CrLf

mov edx, OFFSET prompt
call WriteString
call ReadInt
;mov numberOfCustomers, eax // Deprecated, previous version's code.
mov ecx, eax
call CrLf
```

```
L1:
mov edx, OFFSET prompt2
call WriteString
call ReadInt
mov numberOfDrinks, eax
mov edx, OFFSET prompt3
call WriteString
call ReadChar
mov typeOfDrink, al
call CrLf
mov edx, OFFSET prompt4
call WriteString
call ReadInt
mov numberOfSandwiches, eax
mov edx, OFFSET prompt5
call WriteString
call ReadInt
mov sizeOfSandwiches, eax
mov ebx, numberOfDrinks
.IF typeOfDrink == 'S'
    mov eax, costOfSoda
.ELSE
    mov eax, costOfWater
.ENDIF
mul ebx
mov totalBill, eax
mov ebx, numberOfSandwiches
.IF sizeOfSandwiches == 10
    mov eax, costOfTenInSandwich
.ELSE
    mov eax, costOfTwelveInSandwich
```


okToRegister DWORD ?

textString BYTE "Invalid input detected, program will exit. Enter valid values.", 0

prompt BYTE "Please enter your current INTEGER grade average (0-400): ", 0

prompt2 BYTE "Please enter your current INTEGER amount of credits (0-30): ", 0

failEnding BYTE "FAILURE: The student cannot register.", 0

successfulEnding BYTE "SUCCESS: The student can register.", 0

.code

main proc

mov okToRegister, FALSE

mov edx, OFFSET prompt

call WriteString

call ReadInt

mov gradeAverage, eax

mov ebx, 0

mov edx, 400

cmp eax, ebx

jb FAIL ; If check not pass, go to FAIL (this is jump if below)

cmp eax, edx

ja FAIL ; If check not pass, go to FAIL (this is jump if above)

mov edx, OFFSET prompt2

call WriteString

call ReadInt

mov credits, eax

mov ebx, 0

mov edx, 30

cmp eax, ebx

jb FAIL ; If check not pass, go to FAIL (this is jump if below)

cmp eax, edx

ja FAIL ; If check not pass, go to FAIL (this is jump if above)

jmp SUCCESS

FAIL:

call crlf

mov edx, OFFSET textString

call WriteString

jmp TERMINATE

SUCCESS:

mov eax, gradeAverage

mov ebx, 350 ; .IF gradeAverage > 350 section:

```
cmp eax, ebx
ja END_SUCCESS
```

mov ebx, 250 ; .ELSEIF (gradeAverage > 250) && (credits <= 16) section:

```
cmp eax, ebx
jbe NEXT_IF
```

```
mov eax, credits
mov ebx, 16
```

```
cmp eax, ebx
jbe END_SUCCESS
```

```
NEXT_IF:
mov eax, credits
mov ebx, 12 ; .ELSEIF (credits <= 12) section:
```

```
cmp eax, ebx
jbe END_SUCCESS
jmp END_FAIL
```

END_FAIL: ; If the program is ending in a fail situation (where the student cannot register), then it goes here:

```
call crlf
mov edx, OFFSET failEnding
call WriteString
jmp TERMINATE
```


END_SUCCESS: ; If the program is ending in a success situation, then it goes here:

```
call crlf
mov okToRegister, TRUE
mov edx, OFFSET successfulEnding
call WriteString
jmp TERMINATE
```

TERMINATE: ; Terminate now! The program has effectively ended:
invoke ExitProcess,0

```
main endp
end main
```

```
.IF gradeAverage > 350
    mov OkToRegister,TRUE
.ELSEIF (gradeAverage > 250) && (credits <= 16)
    mov OkToRegister,TRUE
.ELSEIF (credits <= 12)
    mov OkToRegister,TRUE
.ENDIF
```



```
Microsoft Visual Studio Debug Console
Please enter your current INTEGER grade average (0-400): 250
Please enter your current INTEGER amount of credits (0-30): 12
SUCCESS: The student can register.
```

We show that it slides to the final IF statement!

3.

Program Code Provide your code as text, not as a screenshot/image.

Results Please provide your output with a screenshot/image.

3a) $F1 = \sum(0, 1, 5, 7)$

We use the truth table in slides.

A	BC	00	01	11	10
0		1	1	0	0
1		0	1	1	0

$\overline{A}\overline{B}$ $\overline{B}C$ AC

$\overline{A}\overline{B} + \overline{B}C + AC$

~~$\overline{B}(\overline{A} + C) + AC$~~

$\overline{A}\overline{B} + \overline{B}C + AC = F1$

3b) $F2 = \sum(0, 2, 3, 4, 6)$

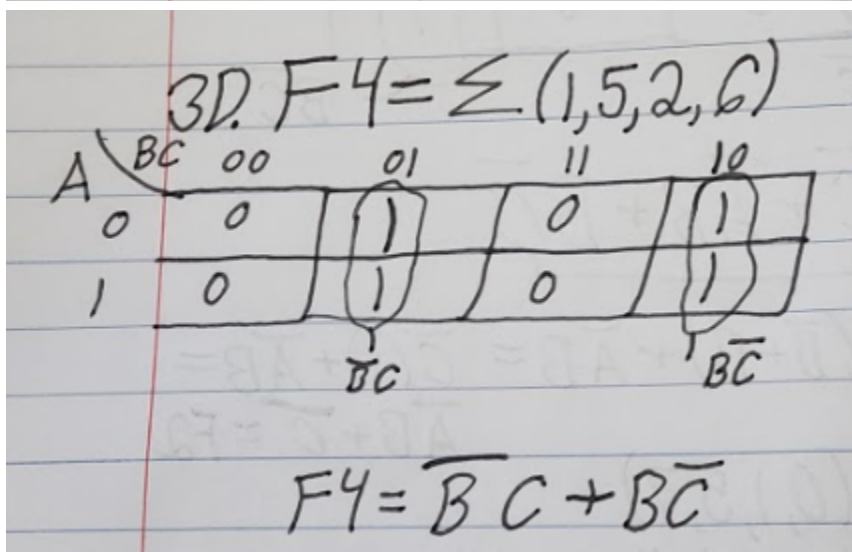
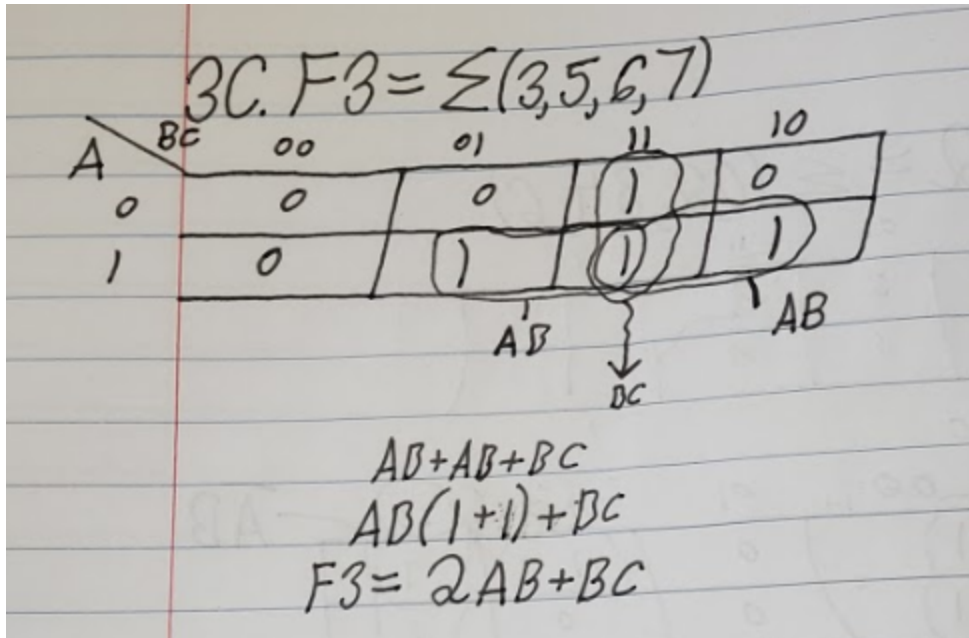
A	BC	00	01	11	10
0		1	0	1	1
1		1	0	0	1

$\overline{B}\overline{C}$ $\overline{A}B$ $B\overline{C}$

$\overline{B}\overline{C} + \overline{A}B + B\overline{C}$

$\overline{C}(\overline{B} + B) + \overline{A}B = \overline{C}(1) + \overline{A}B =$

$\overline{A}B + \overline{C} = F2$



4.

Program Code Provide your code as text, not as a screenshot/image.

Results Please provide your output with a screenshot/image.

INCLUDE Irvine32.inc

.386

.model flat, stdcall

.stack 4096

ExitProcess proto, dwExitCode: dword

.data

n DWORD 0

sum DWORD 0

prompt BYTE "Enter an integer number: ", 0

textString BYTE "Sum ", 0

```

textString2 BYTE "= ", 0
.code
main proc
    call crlf
    mov edx, OFFSET prompt
    call WriteString
    call ReadInt
    mov n, eax

    mov eax, 1 ; i
forLoop:
    mov ebx, sum
    add ebx, eax
    mov sum, ebx

    inc eax
    cmp eax, n ; Compare EAX which at the first run is EAX = 1 (it is the i variable) compared to n
which if you entered 4 would be 4:
    jbe forLoop ; Is i (a.k.a EAX) <= 4 (a.k.a n)? If so keep looping the forLoop, go back to it.
    ; We are finished, now we just need to print out the results:
    call crlf
    mov edx, OFFSET textString
    call WriteString
    mov eax, 1
    call WriteInt
    mov edx, OFFSET textString2
    call WriteString
    mov eax, sum
    call WriteInt
    ; End of program:
    invoke ExitProcess,0
main endp
end main

```



Microsoft Visual Studio Debug Console

```

Enter an integer number: 4
Sum +1= +10

```