LEARNING MANAGEMENT SYSTEM

ALI NADIR	**20K0325**
IMTIAZ ALI	**20K0313**
AHMED ABDULLAH	**20K0470**

Computer Organization and Assembly Language

NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES – FAST

ABSTRACT:

The aim of the project is to enable students in university to effectively manage their academic records, which would primarily consist of records of past assignments, their mid terms and their finals. The application would allow them to see a breakdown of their results for every subject chosen and also will calculate their grade for the chosen subjects. Additionally, students can set and view their own personalized timetable.

INTRODUCTION:

The application allows a student to view and manage academic records for a particular term. It will enable the user to select a certain number of subjects they are currently pursuing, manage academic records for those subjects, and also generate fee challan and transcript for those subjects. Also allows for setting and viewing timetable.

LITERATURE REVIEW:

LMSs play an important role in enhancing and facilitating teaching and learning. LMSs not only enable the delivery of instructions and electronic resources to improve and augment student learning in a collaborative environment, but also allow instructors to focus on designing meaningful pedagogical activities (Kattoua, Al-Lozi, and Alrowwad 2016). It should be noted that most LMS software available relies on online connectivity, and for good reason as well. But this should not negate the fact that an offline, almost instantaneous solution to every students' academic record keeping should exist, and this project is driven primarily by the motivation. LMS software such as those based on Sakai, such as SLATE and FLEX offer students to create their own timetables and study plans which is why this application models the same.

PROBLEM DEFINITION:

Set, manage, view, and extract academic records as efficiently and intuitively as possible, and allow timetable functionality.

METHODOLOGY:

Procedures have been greatly used to reduce structural complexity of the program and to enhance readability. Due to the scope of the language used, imperative programming approach has been adopted. Since user input is key to the application, data validation

has been implemented in as many areas as possible. Conditional statements, loop structures and conditional breaks play an integral part in the functionality of the program. Nested function calls have been used. Conditional jumps have been implemented as well. Since most of the loops used in the code exceed the maximum memory that can be allocated, flag jumps have been used as well. In addition, the program is heavily reliant on local variables in most of the procedures used.

DETAILED DESIGN AND ARCHITECTURE:

Program is divided into various subsystems or routines that drive the program. The flow is as follows:

- Login with valid student ID
- Once, redirect to function that sets your marks
- Once validated marks successfully registered in system, display menu
- Menu consists of a total of 8 procedures that is continuously displayed and clears the screen.
- Procedures within Menu procedure:
    - Display subjects
    - Set Marks
    - View academic records
    - Generate fee challan
    - View transcript
    - Set timetable
    - View timetable
    - Logout
- These functions contain further subroutines such as grade subject, calculate total, display grade, check marks etc. which are utilized in nested function calls by the menu procedures

IMPLEMENTATION, TESTING, AND PROGRAMMING CODE:

FUNCTIONALITIES:

- Procedure Menu
    - **Driver function in Main Proc**
    - Displays/prints all the available features which are part of the program on console
    - Takes input from user to get choice of feature
    - Simulates SWITCH conditional structure to jump to relevant conditional block for valid input
    - Each block calls a particular subroutine
    - Clears screen each time it is called

- Procedure SetStudentID
    - Fetches Student ID from user as string using readchar procedure
    - After input, nested call to CheckID is made with string id passed as stack parameter
    - ID is validated based on format **XXYXXXX** where:
        - **X** is a digit
        - **Y** is an alphabetical character
    - If valid, update variable valid1 with value 1, else mov valid1,0
    - Loop if invalid input, or exit program

- Procedure SetSubjects
    - Asks user to enter which year of academic calendar/term they are in
    - Upon getting input, display relevant Subject list
    - List created for each year
    - **List is a 2D array of characters**, so it is essentially an **array of STRINGS**
    - Update array mysubjects with chosen subject
    - HOW ARRAY MYSUBJECTS WORKS:
        - Throughout the program, mysubjects is used to display the **strings** of the chosen 6 subjects

        - **This is implemented by storing the offset of a row of the 2d array of a particular year (representing a null terminated string), into the 1D array of DWORD type MySubjects**
        - **This offset is stored in the memory location pointing to an element of array MySubjects**
        - **In order to read chosen subjects string, simply:**
            - **Move the value withing array MySubjects into EDX register**
            - **Now call writestring procedure to display the null terminated string of chosen subject**
    - Do this for all 6 subjects then return control to driver program Main

- Procedure DisplaySubjects
    - Displays the strings of the selected subjects using technique mentioned in documentation of SetSubjects
    - Used in other user defined procedures to display subjects for input/selection
    - Usually used when user input is expected

- Procedure SetMarks
    - Used to retrieve marks for a particular subject
    - Updates all the records for a subject including Assigment 1 marks Assignment 2 marks, Mid 1 and 2 marks, and Final marks for each subject chosen
- Procedure ViewRecords
    - Does two things:
        - **VIEW RECORDS**
            - Shows user their academic record of the selected subject in marks
            - Shows user their subject breakdown in terms of weightage composition summing to 100%
        - **CALCULATE SUBJECT GPA**
            - Make nested call to procedure GradeSub to obtain letter grade for the subject
- Procedure GetChallan
    - Requests user to enter number of credit hours for each subject
    - Calculates total cost for all subjects
    - Displays total calculated cost with composition on console

- Procedure GetTranscript
    - Displays the users SGPA by averaging the GPAs of each chosen subject
    - Uses MUL and DIV instructions
    - Makes call to subroutine GetSGPA
        - Procedure GetSGPA
            - Uses formula SGPA = (Sum of (Subject GPA * Credit hours)/Total credit hours attempted

- Procedure SetTable
    - Uses 2D array arr1 which basically represents data definition of the timetable
    - This timetable will contain the values of the indexes of **array mysubjects**
    - Arr1 stores values range from 0 to 5 (6 subjects)
    - These indexes are fetched into index register **esi or edi**
    - Retrieved index, currently in **esi/edi**, will now be used to resolve the offset of stored string within array **mysubjects**
    - **1 (Retrieve index from 2D array Arr1)**
        - **Mov eax,[ebx+edi]**
    - **2 (Resolve offset/address of string stored in 1D Array MySubjects)**
        - **Mov esi,eax**
        - **Mov edx,MySubjects[esi*TYPE MySubjects]**

- Procedure ViewTable
    - Input is taken for every day of the week
    - For every day, there are 8 slots as shown below, so input taken 8 times per day
    - Choice of either break or class is given
    - If class is selected, user enters choice of subject to be added to slot
    - If break selected, index manipulated in such a way that table displays "FREE" on console
    - For both breaks and a certain subject, a maximum of three breaks and maximum of three slots of the "SAME SUBJECT" are allowed
    - Once timetable is set for all days, table is then displayed
    - For display, ViewTable procedure can be called
    - ViewTable procedure offers user 2 options:
        - Display WHOLE time table
        - Display CURRENT class
    - **CURRENT**

- Uses Kernel32.lib prototyped procedure GetLocalTime to store system time data in the form of a structure of name LPSYSTEMTIME
- Attributes of **LPSYSTEMTIME** used to obtain:
    - **Row index** is value of struct member wHours, since each row of timetable represents a slot between two hours
    - **wHours** contains values from 0 to 24 (24 hr format)
    - Lower hour used for indexing rows
    - wHours validated and 8 subtracted to normalize for row manipulation of 2D array
    - **Column Index** is value of struct member wDayofWeek which contains values from 1(Monday) to 7 (Sunday)

- **Black Box testing technique used along with debugging for exploring code output generation and Runtime error resolution**
- **Program uses console to interact with user for input and I/O**

LIBRARIES USED:

- Irvine32.inc
    - For basic console I/O
    - Getting user input
    - Comparisons using cmp instructions
    - Conditional jumps and loop structures
    - Clearing the screen and more
- Kernel32.lib
    - Included with directive “includelib”
    - Primarily serves the purpose to enable use of GetLocalTime procedure
    - Used to obtain prototype of the function GetLocalTime
    - Used to retrieve local system time in defined structure LPSYSTEMTIME
- Macros.inc
    - Used for mwrite instructions for displaying string output on console

CODE:

```
INCLUDE Irvine32.inc
includelib kernel32.lib
include macros.inc


.data
arr1 dword 5 dup(6)
rowsize=($-arr1)
dword 5 dup(6)
dword 5 dup(6)
dword 5 dup(6)
dword 5 dup(6)
dword 5 dup(6)
dword 5 dup(6)
dword 5 dup(6)


days byte "Monday   ",0
daysize=($-days)
     byte "Tuesday  ",0
     byte "Wednesday",0
     byte "Thursday ",0
     byte "Friday   ",0
tempstr byte 10 dup(0)

LPSYSTEMTIME STRUCT
```

```
    wYear       WORD ?
    wMonth      WORD ?
    wDayOfWeek  WORD ?
    wDay        WORD ?
    wHour       WORD ?
    wMinute     WORD ?
    wSecond     WORD ?
    wMilliseconds WORD ?
    wDate       WORD ?
LPSYSTEMTIME ENDS

localt LPSYSTEMTIME <>

maxlength=50
valid dword 0
id byte 10 dup(?)
size1 dword ?
duplicate dword 6 dup(?)
valid1 dword 0
yearchoice dword ?
locked1 dword 0
crarr dword 6 dup(?)
overwrite1 dword 0
gotcredits dword 0
;*******************2d array for timetable******************
table1 byte 5 dup(6)
ftrowsize=($-table1)
```

```
colsize=lengthof table1
byte 5 dup(6)
byte 5 dup(6)
byte 5 dup(6)
byte 5 dup(6)
byte 5 dup(6)
byte 5 dup(6)
byte 5 dup(6)
;********************2d array for timetable******************

;********************array for chosen subjects******************
;*********6 subjects to be selected**********
;**************array of subject strings***************
mysubjects dword 6 dup(?)
subflag dword 6 dup(0)
subgpa dword 6 dup(0)

sub1 dword 5 dup(?)
sub2 dword 5 dup(?)
sub3 dword 5 dup(?)
sub4 dword 5 dup(?)
sub5 dword 5 dup(?)
sub6 dword 5 dup(?)
;********************2d array for chosen subjects******************
```

```
;*******************2d       array       for       1st       yr
strings******************
;***********lengths kept equal
first byte "English Composition and Comprehension  ",0
firstrowsize=($-first)
firstsize=11
   byte       "Applied Physics                       ",0
   byte       "Islamiyat                             ",0
   byte    "Programming Fundamentals               ",0
   byte    "Calculus                               ",0
   byte    "Information and Computer Technology    ",0
   byte    "Object Oriented Programming            ",0
   byte    "Communication and Presentation Skills  ",0
   byte    "Digital Logic Design                   ",0
   byte    "Pakistan Studies                       ",0
   byte    "Differential Equations                 ",0
;*******************2d       array       for       1st       yr
strings******************

;*******************2d array for 2nd strings******************
second byte "Data Structures                     ",0
secondrowsize=($-second)
secondsize=11
   byte        "Computer Organization Assembly Lang.",0
   byte        "Discrete Structures                 ",0
   byte     "Linear Algebra                     ",0
   byte     "Fundamentals Of Management         ",0
```

```
   byte     "Psychology                          ",0
   byte     "Sociology                           ",0
   byte     "DataBase Systems                    ",0
   byte     "Operating Systems                   ",0
   byte     "Design and Analysis of Algorithms   ",0
   byte     "Probability & Statistics            ",0
;*******************2d array for 2nd strings******************

;*******************2d array for 3rd strings******************
third byte  "Theory of Automata                  ",0
thirdrowsize=($-third)
thirdsize=11
   byte        "Computer Networks                   ",0
   byte        "Automated Systems                   ",0
   byte     "Probabilistic Models                ",0
   byte     "Software Design and Analysis        ",0
   byte     "Technical & Business Writing        ",0
   byte     "Numerical Computing                 ",0
   byte     "Software Engineering                ",0
   byte     "Parallel & Distributed Computing    ",0
   byte     "Artificial Intelligence             ",0
   byte     "BlockChain                          ",0
;*******************2d array for 3rd strings******************

;*******************2d array for 4th strings******************

fourth byte "Final Year Project-I                ",0
```

```
fourthrowsize=($-fourth)
fourthsize=11
   byte         "Information Security               ",0
   byte         "Professional Practices             ",0
   byte         "Web Development                    ",0
   byte         "Hardware Design                    ",0
   byte         "Final Year Project-II              ",0
   byte         "Ethical Hacking                    ",0
   byte         "Data Visualization                 ",0
   byte         "Data Analysis                      ",0
   byte         "Robotics                           ",0
   byte         "Organizational Management          ",0
;*******************2d array for 4th strings******************

;***************arrays                for                subject
marks************************

;***************arrays                for                subject
marks************************


.code
main PROC
o1:
call clrscr
push offset id
call login
```

```
mov eax,0
cmp valid,eax
jne out1
call crlf
mwrite "ID entered is invalid. Enter 1 to try again, or 2 to
exit: "
call readint
cmp eax,2
je e2
loop o1

out1:
call crlf
mwrite "Welcome user "
mov edx,offset id
call writestring

;display menu only after subjects have been entered
push offset mysubjects
call setsubjects

mwrite <0ah,0dh,"Enter any key to proceed...">
push eax
call readchar
pop eax

;loop until user logs out
```

```
kl1:
call clrscr
call displaymenu
call readint
cmp eax,8
je e2
cmp eax,1
je case1
cmp eax,2
je case2
cmp eax,3
je case3
cmp eax,4
je case4
cmp eax,5
je case5
cmp eax,6
je case6
cmp eax,7
je case7

case1:
call displaysubs
mwrite <"Press any key to continue...">
call readchar
jmp kl1
case2:
```

```
call setmarks
mov overwrite1,0
jmp kl1
case3:
call viewrecords
jmp kl1
case4:
call getchallan
jmp kl1
case5:
call gettranscript
jmp kl1
case6:
push offset mysubjects
push offset arr1
call settable
jmp kl1
case7:
mwrite <0ah,0dh,"Viewing table">
call readchar
call clrscr
call viewtable
jmp kl1
;
e2:
call crlf
mwrite "Now exiting..."
```

```
exit
main endp

gettranscript proc uses eax edx ecx esi

mov eax,gotcredits
cmp eax,0
je invld

mov ecx,6
mov esi,0
loo1:
mov eax,subflag[esi*type subflag]
cmp eax,0
je invld
mov eax,subgpa[esi*type subgpa]
cmp eax,0
je invld
inc esi
loop loo1

mwrite <0ah,0dh,"Warning: Generating transcript will lock your
grades in the system. Enter 1 to continue, any other key to
return: ">
call readint
cmp eax,1
jne scc
```

```
mov locked1,1

mov eax,250
call delay
mwrite <0ah,0dh,"Now generating your transcript...">
mov eax,250
call delay

mov esi,0
mov edi,6
mov ecx,6

call crlf
call crlf
mwrite <0ah,0dh,"TRANSCRIPT FOR STUDENT ">
mov edx,offset id
call writestring
mwrite <0ah,0dh,"YEAR ">
mov eax,yearchoice
call writedec

call crlf
call crlf
call crlf
mwrite <0ah,0dh,"||                Subject                |  GPA
|  Grade  ||">
```

```
itlp1:
mwrite "  "
mov edx,mysubjects[esi*type mysubjects]
call writestring
mwrite "  "
mov eax,subgpa[esi*type subgpa]
call writedec
mwrite ".00    "
cmp eax,4
je apluss
cmp eax,3
je ab1
cmp eax,2
je cd1
cmp eax,1
je fa1

apluss:
mwrite <"A/+",0ah,0dh>
jmp sdf1

ab1:
mwrite <"A/B",0ah,0dh>
jmp sdf1

cd1:
mwrite <"C/D",0ah,0dh>
```

```
jmp sdf1

fa1:
mwrite <"F",0ah,0dh>
jmp sdf1
sdf1:
inc esi
sub edi,1
jnz itlp1

breaklp:

call getsgpa

call crlf
call crlf
mov eax,250
call delay
mwrite <"Transcript Generated.">
jmp scc

invld:
mwrite <0ah,0dh,"Error. Could not generate transcript. Marks not
entered completely, GPA not yet calculated, or Fee challan not
generated.">
jmp scc
```

```
scc:
mwrite <0ah,0dh,"Press any key to continue...">
call readchar
ret
gettranscript endp

getsgpa proc uses eax esi
local onecount:dword,twocount:dword
mov twocount,0
mov onecount,0
mov esi,0
mov ecx,6
ulp1:
mov eax,subgpa[esi*type subgpa]
mov ebx,crarr[esi*type crarr]
add twocount,ebx
mul ebx
add onecount,eax
inc esi
loop ulp1

mov ebx,twocount
mov eax,onecount
cdq
div ebx
```

```
mwrite <0ah,0dh,"Over GPA for this academic year (SGPA) is: ">
call writedec
call crlf
call crlf
ret
getsgpa endp

getchallan proc uses esi edi eax ebx ecx edx
local stotal:dword
;crarr[6]:dword
call getcreditcost;cost in eax
push eax

mov stotal,0
mov esi,offset mysubjects
mov edi,0
mov ecx,6

la1:
mwrite <0ah,0dh,"Enter credit hours for ">
mov edx,[esi]
call writestring
mwrite ": "
call readint
mov crarr[edi*type crarr],eax
add esi,4
inc edi
```

```
loop la1

pop eax;cost in eax
mov ebx,eax
mov esi,offset mysubjects
mov edi,0
mov ecx,6
call crlf
call crlf
mwrite<"Now printing fee challan...">
call crlf
call crlf
mwrite<"*********************************FEE CHALLAN FOR USER ">
mov edx,offset id
call writestring
mwrite<"*********************************">

call crlf
call crlf
lla2:
mwrite <0ah,0dh,"For ">
mov edx,[esi]
call writestring
mwrite <": ",0ah,0dh>
mwrite <"Credit Hours: ">
mov eax,crarr[edi*type crarr]
call writedec
```

```
push eax
mwrite <0ah,0dh,"Cost per Credit Hour: PKR ">
mov eax,ebx
call writedec
pop eax
mul ebx
add stotal,eax
add esi,4
inc edi
call crlf
loop lla2
mov gotcredits,1
mwrite "*****************Total charges due for the 6 chosen
subjects: PKR "
mov eax,stotal
call writedec
mwrite<"*****************",0ah,0dh,"***************************
*************************************************************
****",0ah,0dh>
call crlf
mwrite <"Press any key to continue...">
call readchar
ret
getchallan endp

getcreditcost proc uses ebx
mov ebx,1
cmp yearchoice,ebx
```

```
je yr1
inc ebx
cmp yearchoice,ebx
je yr2
inc ebx
cmp yearchoice,ebx
je yr3
inc ebx
cmp yearchoice,ebx
je yr4

yr1:
mov eax,5000
jmp r1
yr2:
mov eax,6500
jmp r1
yr3:
mov eax,7500
jmp r1
yr4:
mov eax,10000
jmp r1

r1:ret
getcreditcost endp
```

```
viewrecords proc
local tempcount:dword

retry1:
mwrite <0ah,0dh,"Select subject whose record you wish to view:
">
call displaysubs
mwrite <0ah,0dh,"Enter choice: ">
call readint
cmp eax,1
jb retry1
cmp eax,6
ja retry1
mov tempcount,eax
cmp eax,1
je onearr
cmp eax,2
je twoarr
cmp eax,3
je threearr
cmp eax,4
je fourarr
cmp eax,5
je fivearr
cmp eax,6
je sixarr
```

```
onearr:
mov esi,0
cmp subflag[esi*type subflag],0
je err11
mov esi,offset sub1
jmp show1

twoarr:
mov esi,1
cmp subflag[esi*type subflag],0
je err11
mov esi,offset sub2
jmp show1

threearr:
mov esi,2
cmp subflag[esi*type subflag],0
je err11
mov esi,offset sub3
jmp show1

fourarr:
mov esi,3
cmp subflag[esi*type subflag],0
je err11
mov esi,offset sub4
jmp show1
```

```
fivearr:
mov esi,4
cmp subflag[esi*type subflag],0
je err11
mov esi,offset sub5
jmp show1

sixarr:
mov esi,5
cmp subflag[esi*type subflag],0
je err11
mov esi,offset sub6
jmp show1

show1:
push esi
mwrite <0dh,0ah,"|| Assignment 1 | Assignment 2 |     Mid 1     |
Mid 2     |     Final     ||",0ah,0dh>
mov ecx,5
mwrite <"|">
lpp2:
mwrite <"|     ">
mov eax,[esi]
call writedec
mwrite <"        ">
add esi,4
```

```
loop lpp2
mwrite <"||">
pop esi

mov ebx,0
mwrite <0ah,0dh,"Press any key to view weightage breakdown.">
mwrite <0dh,0ah,"|| Assignment 1 | Assignment 2 |     Mid 1     |
Mid 2     |     Final     ||",0ah,0dh>
call readchar
mov eax,[esi]
mov ecx,10
mul ecx
mov ecx,20
cdq
div ecx
add ebx,eax
mwrite <"||     ">
call writedec
mwrite <".0 %   ">
add esi,4
mov eax,[esi]
mov ecx,10
mul ecx
mov ecx,20
cdq
div ecx
add ebx,eax
```

```
mwrite <"|     ">
call writedec
mwrite <".0 %   ">
add esi,4
mov eax,[esi]
mov ecx,15
mul ecx
mov ecx,40
cdq
div ecx
add ebx,eax
mwrite <"|     ">
call writedec
mwrite <".0 %   ">
add esi,4
mov eax,[esi]
mov ecx,15
mul ecx
mov ecx,40
cdq
div ecx
add ebx,eax
mwrite <"|     ">
call writedec
mwrite <".0 %   ">

add esi,4
```

```
mov eax,[esi]
mov ecx,50
mul ecx
mov ecx,100
cdq
div ecx
add ebx,eax
mwrite <"|     ">
call writedec
mwrite <".0 %   ">

mwrite <0ah,0dh,"Grand Total for subject ">
mov esi,tempcount
dec esi
mov edx,mysubjects[esi*type mysubjects]
call writestring
mwrite <" is: ">
mov eax,ebx
call writedec

mwrite <0ah,0dh,"Overall grade:  ">
push eax
call gradesub
jmp succ1

err11:
```

```
mwrite <0ah,0dh,"Error: Marks have not been completely entered.
Try again later.",0ah,0dh>


succ1:

mwrite <0ah,0dh,"Press any key to continue...">

call readchar

ret

viewrecords endp


gradesub proc uses eax,

varr1:dword

mov eax,varr1

cmp eax,90

jae aplus

cmp eax,80

jae aonly

cmp eax,70

jae bonly

cmp eax,60

jae conly

cmp eax,50

jae donly

jmp lowgrade


aplus:

mov eax,"A"

call writechar
```

```
mov eax,"+"
call writechar
mwrite <0ah,0dh,"Subject GPA: 4">
mov eax,4
mov subgpa[esi*type subgpa],eax
jmp end1

aonly:
mov eax,"A"
call writechar
mwrite <0ah,0dh,"Subject GPA: 4">
mov eax,4
mov subgpa[esi*type subgpa],eax
jmp end1

bonly:
mov eax,"B"
call writechar
mwrite <0ah,0dh,"Subject GPA: 3.66">
mov eax,3
mov subgpa[esi*type subgpa],eax
jmp end1

conly:
mov eax,"C"
call writechar
mwrite <0ah,0dh,"Subject GPA: 3.33">
```

```
mov eax,3
mov subgpa[esi*type subgpa],eax
jmp end1

donly:
mov eax,"D"
call writechar
mwrite <0ah,0dh,"Subject GPA: 2.66">
mov eax,2
mov subgpa[esi*type subgpa],eax
jmp end1

lowgrade:
mov eax,"F"
call writechar
mwrite <0ah,0dh,"Subject GPA: 1.00">
mov eax,1
mov subgpa[esi*type subgpa],eax
jmp end1

end1:
ret
gradesub endp

setmarks proc uses eax edx ecx ebx esi
local ch1:dword
```

```
cmp locked1,1
je invd3

tryag:
mwrite <0ah,0dh,"Select subject whose marks are to be entered:
">
call displaysubs
mwrite <0ah,0dh,"Enter choice: ">
call readint

cmp eax,1
jb try2
cmp eax,6
ja try2

jmp prcd

try2:
mwrite <"Invalid choice entered.",0ah,0dh>
jmp tryag

prcd:
mov esi,eax
dec esi
mov ch1,esi
cmp eax,1
je sb1
```

```
cmp eax,2
je sb2
cmp eax,3
je sb3
cmp eax,4
je sb4
cmp eax,5
je sb5
cmp eax,6
je sb6

sb1:
mov esi,ch1
mov eax,1
cmp overwrite1,eax
je sb12
mov eax,1
cmp subflag[esi*type subflag],eax
je er1

sb12:
mwrite <0ah,0dh,"Enter assignment 1 marks: ">
call readint
mov edx,1
push edx
call checknum
push eax
```

```
mov eax,0
cmp valid1,eax
pop eax
je sb1

mov esi,0
mov sub1[esi*type sub1],eax

mwrite <0ah,0dh,"Enter assignment 2 marks: ">
call readint
mov edx,1
push edx
call checknum
push eax
mov eax,0
cmp valid1,eax
pop eax
je sb1

inc esi
mov sub1[esi*type sub1],eax

mwrite <0ah,0dh,"Enter Mid 1 marks: ">
call readint
mov edx,2
push edx
call checknum
```

```
push eax
mov eax,0
cmp valid1,eax
pop eax
je sb1

inc esi
mov sub1[esi*type sub1],eax

mwrite <0ah,0dh,"Enter Mid 2 marks: ">
call readint
mov edx,2
push edx
call checknum
push eax
mov eax,0
cmp valid1,eax
pop eax
je sb1

inc esi
mov sub1[esi*type sub1],eax

mwrite <0ah,0dh,"Enter Final Exam marks: ">
call readint
mov edx,3
push edx
```

```
call checknum
push eax
mov eax,0
cmp valid1,eax
pop eax
je sb1

inc esi
mov sub1[esi*type sub1],eax
mov esi,ch1
mov subflag[esi*type subflag],1
jmp success

sb2:
mov esi,ch1
mov eax,1
cmp overwrite1,eax
je sb22
mov eax,1
cmp subflag[esi*type subflag],eax
je er1

sb22:
mwrite <0ah,0dh,"Enter assignment 1 marks: ">
call readint
mov edx,1
push edx
```

```
call checknum
push eax
mov eax,0
cmp valid1,eax
pop eax
je sb2

mov esi,0
mov sub2[esi*type sub2],eax

mwrite <0ah,0dh,"Enter assignment 2 marks: ">
call readint
mov edx,1
push edx
call checknum
push eax
mov eax,0
cmp valid1,eax
pop eax
je sb2

inc esi
mov sub2[esi*type sub2],eax

mwrite <0ah,0dh,"Enter Mid 1 marks: ">
call readint
mov edx,2
```

```
push edx
call checknum
push eax
mov eax,0
cmp valid1,eax
pop eax
je sb2

inc esi
mov sub2[esi*type sub2],eax

mwrite <0ah,0dh,"Enter Mid 2 marks: ">
call readint
mov edx,2
push edx
call checknum
push eax
mov eax,0
cmp valid1,eax
pop eax
je sb2

inc esi
mov sub2[esi*type sub2],eax

mwrite <0ah,0dh,"Enter Final Exam marks: ">
call readint
```

```
mov edx,3
push edx
call checknum
push eax
mov eax,0
cmp valid1,eax
pop eax
je sb2

inc esi
mov sub2[esi*type sub2],eax
mov esi,ch1
mov subflag[esi*type subflag],1
jmp success

sb3:
mov esi,ch1
mov eax,1
cmp overwrite1,eax
je sb32
mov eax,1
cmp subflag[esi*type subflag],eax
je er1

sb32:

mwrite <0ah,0dh,"Enter assignment 1 marks: ">
```

```
call readint
mov edx,1
push edx
call checknum
push eax
mov eax,0
cmp valid1,eax
pop eax
je sb3

mov esi,0
mov sub3[esi*type sub3],eax

mwrite <0ah,0dh,"Enter assignment 2 marks: ">
call readint
mov edx,1
push edx
call checknum
push eax
mov eax,0
cmp valid1,eax
pop eax
je sb3

inc esi
mov sub3[esi*type sub3],eax
```

```
mwrite <0ah,0dh,"Enter Mid 1 marks: ">
call readint
mov edx,2
push edx
call checknum
push eax
mov eax,0
cmp valid1,eax
pop eax
je sb3

inc esi
mov sub3[esi*type sub3],eax

mwrite <0ah,0dh,"Enter Mid 2 marks: ">
call readint
mov edx,2
push edx
call checknum
push eax
mov eax,0
cmp valid1,eax
pop eax
je sb3

inc esi
mov sub3[esi*type sub3],eax
```

```
mwrite <0ah,0dh,"Enter Final Exam marks: ">
call readint
mov edx,3
push edx
call checknum
push eax
mov eax,0
cmp valid1,eax
pop eax
je sb3

inc esi
mov sub3[esi*type sub3],eax
mov esi,ch1
mov subflag[esi*type subflag],1
jmp success

sb4:
mov esi,ch1
mov eax,1
cmp overwrite1,eax
je sb42
mov eax,1
cmp subflag[esi*type subflag],eax
je er1
```

```
sb42:

mwrite <0ah,0dh,"Enter assignment 1 marks: ">
call readint
mov edx,1
push edx
call checknum
push eax
mov eax,0
cmp valid1,eax
pop eax
je sb4

mov esi,0
mov sub4[esi*type sub4],eax

mwrite <0ah,0dh,"Enter assignment 2 marks: ">
call readint
mov edx,1
push edx
call checknum
push eax
mov eax,0
cmp valid1,eax
pop eax
je sb4
```

```
inc esi
mov sub4[esi*type sub4],eax

mwrite <0ah,0dh,"Enter Mid 1 marks: ">
call readint
mov edx,2
push edx
call checknum
push eax
mov eax,0
cmp valid1,eax
pop eax
je sb4

inc esi
mov sub4[esi*type sub4],eax

mwrite <0ah,0dh,"Enter Mid 2 marks: ">
call readint
mov edx,2
push edx
call checknum
push eax
mov eax,0
cmp valid1,eax
pop eax
je sb4
```

```
inc esi
mov sub4[esi*type sub4],eax

mwrite <0ah,0dh,"Enter Final Exam marks: ">
call readint
mov edx,3
push edx
call checknum
push eax
mov eax,0
cmp valid1,eax
pop eax
je sb4

inc esi
mov sub4[esi*type sub4],eax
mov esi,ch1
mov subflag[esi*type subflag],1
jmp success

sb5:
mov esi,ch1
mov eax,1
cmp overwrite1,eax
je sb52
mov eax,1
```

```
cmp subflag[esi*type subflag],eax
je er1

sb52:

mwrite <0ah,0dh,"Enter assignment 1 marks: ">
call readint
mov edx,1
push edx
call checknum
push eax
mov eax,0
cmp valid1,eax
pop eax
je sb5

mov esi,0
mov sub5[esi*type sub5],eax

mwrite <0ah,0dh,"Enter assignment 2 marks: ">
call readint
mov edx,1
push edx
call checknum
push eax
mov eax,0
cmp valid1,eax
```

```
pop eax
je sb5

inc esi
mov sub5[esi*type sub5],eax

mwrite <0ah,0dh,"Enter Mid 1 marks: ">
call readint
mov edx,2
push edx
call checknum
push eax
mov eax,0
cmp valid1,eax
pop eax
je sb5

inc esi
mov sub5[esi*type sub5],eax

mwrite <0ah,0dh,"Enter Mid 2 marks: ">
call readint
mov edx,2
push edx
call checknum
push eax
mov eax,0
```

```
cmp valid1,eax
pop eax
je sb5

inc esi
mov sub5[esi*type sub5],eax

mwrite <0ah,0dh,"Enter Final Exam marks: ">
call readint
mov edx,3
push edx
call checknum
push eax
mov eax,0
cmp valid1,eax
pop eax
je sb5

inc esi
mov sub5[esi*type sub5],eax
mov esi,ch1
mov subflag[esi*type subflag],1

jmp success

sb6:
mov esi,ch1
```

```
mov eax,1
cmp overwrite1,eax
je sb62
mov eax,1
cmp subflag[esi*type subflag],eax
je er1

sb62:

mwrite <0ah,0dh,"Enter assignment 1 marks: ">
call readint
mov edx,1
push edx
call checknum
push eax
mov eax,0
cmp valid1,eax
pop eax
je sb6

mov esi,0
mov sub6[esi*type sub6],eax

mwrite <0ah,0dh,"Enter assignment 2 marks: ">
call readint
mov edx,1
push edx
```

```
call checknum
push eax
mov eax,0
cmp valid1,eax
pop eax
je sb6

inc esi
mov sub6[esi*type sub6],eax

mwrite <0ah,0dh,"Enter Mid 1 marks: ">
call readint
mov edx,2
push edx
call checknum
push eax
mov eax,0
cmp valid1,eax
pop eax
je sb6

inc esi
mov sub6[esi*type sub6],eax

mwrite <0ah,0dh,"Enter Mid 2 marks: ">
call readint
mov edx,2
```

```
push edx
call checknum
push eax
mov eax,0
cmp valid1,eax
pop eax
je sb6

inc esi
mov sub6[esi*type sub6],eax

mwrite <0ah,0dh,"Enter Final Exam marks: ">
call readint
mov edx,3
push edx
call checknum
push eax
mov eax,0
cmp valid1,eax
pop eax
je sb6

inc esi
mov sub6[esi*type sub6],eax
mov esi,ch1
mov subflag[esi*type subflag],1
```

```
jmp success

er1:
mwrite <0ah,0dh,"Chosen subject's marks have already been entered. ">
mwrite <0ah,0dh,"Do you wish to overwrite subject marks? Enter 1 if yes, else for no: ">
call readint
cmp eax,1
je trr1
jmp ex1
trr1:
mov overwrite1,1
jmp tryag
jmp ex1

invd3:
mwrite <0ah,0dh,"Marks have now been locked in the system. Can not proceed">
jmp ex1

success:
mwrite <0ah,0dh,"Chosen subject's marks successfuly entered.">

ex1:
mwrite <0ah,0dh,"Press any key to continue.">
call readchar
ret
```

```
setmarks endp

checknum proc uses edx eax,
var1:dword
local lcount:dword

mov ecx,1
cmp var1,ecx
je checkas
inc ecx
cmp var1,ecx
je checkmid
inc ecx
cmp var1,ecx
je checkf

checkas:
cmp eax,0
jb invalid2
cmp eax,20
ja invalid2
mov valid1,1
jmp scc

checkmid:
cmp eax,0
jb invalid2
```

```
cmp eax,40
ja invalid2
mov valid1,1
jmp scc

checkf:
cmp eax,0
jb invalid2
cmp eax,100
ja invalid2
mov valid1,1
jmp scc

invalid2:
mov valid1,0
mwrite   <0ah,0dh,"Incorrect   entry.   Please   enter   marks
again.",0ah,0dh>
scc:
ret
checknum endp

displaysubs proc uses esi eax ecx edx
local count1:dword
mov count1,0
mov ecx,lengthof mysubjects
mov esi,0
call crlf
```

```
inlp:
inc count1
mov eax,count1
call writedec
mwrite ". "
mov edx,mysubjects[esi*type mysubjects]
call writestring
call crlf
inc esi
loop inlp
;mwrite <"Press any key to continue...">
;call readchar
ret
displaysubs endp

setsubjects proc uses eax ecx edx ebx esi,
var1:ptr dword
local countvar: dword
;call clrscr
mov countvar,0
mov esi,var1
mov edi,0

lp1:
mwrite <0ah,0dh,"Please select which year you are in: ">
mwrite <0ah,0dh,"1. First">
mwrite <0ah,0dh,"2. Second">
```

```
mwrite <0ah,0dh,"3. Third">
mwrite <0ah,0dh,"4. Fourth">
mwrite <0ah,0dh,"Enter choice: ">
call readint
mov yearchoice,eax
cmp eax,1
je year1
cmp eax,2
je year2
cmp eax,3
je year3
cmp eax,4
je year4
mwrite <0ah,0dh,"Invalid entry, try again: ">
jmp lp1

year1:
mov ebx, offset first
mov eax,firstrowsize
mov size1,eax
jmp set
year2:
mov ebx, offset second
mov eax,secondrowsize
mov size1,eax
jmp set
year3:
```

```
mov ebx, offset third
mov eax,thirdrowsize
mov size1,eax
jmp set
year4:
mov ebx, offset fourth
mov eax,fourthrowsize
mov size1,eax

set:
mwrite <0ah,0dh,"Please select any 6 subjects: ",0ah,0dh>

mov ecx,firstsize
mov eax,0

push ebx
lp2:
inc eax
call writedec
mwrite ". "
mov edx,ebx
call writestring
add ebx,size1
call crlf
loop lp2
pop ebx
```

```
try:
mov eax,6
cmp countvar,eax
je outofloop1
inc countvar
mov edx,edi
tryone:
mwrite <0ah,0dh,"Enter your choice: ">
call readint
cmp eax,11
ja nvd3
cmp eax,0
jbe nvd3
jmp fine11

nvd3:
mwrite <"Invalid choice entered.">
jmp tryone

fine11:

mov ecx,countvar
dec ecx
cmp ecx,0
je moveon
```

```
mov edi,0
checkingl:
cmp eax,duplicate[edi*type duplicate]
je nvd3
inc edi
loop checkingl
mov edi,edx

moveon:
mov duplicate[edi*type duplicate],eax
inc edi

mov ecx,firstsize
mov edx,0
push ebx

lp3:
inc edx
cmp edx,eax
je outofloop
add ebx,size1
loop lp3

outofloop:
mov [esi],ebx;offset of string stored in mysubjects array
add esi,4
```

```
pop ebx
jmp try

outofloop1:
mwrite <0ah,0dh,"Subjects successfully stored. You have chosen:
",0ah,0dh>
mov esi,var1
mov ecx,6
mov countvar,0
dploop:
inc countvar
mov eax,countvar
call writedec
mwrite ". "
mov edx,[esi]
call writestring
call crlf
add esi,4
loop dploop

ret
setsubjects endp

displaymenu proc
mwrite <"Please select one of the following: ",0ah,0dh>
mwrite <"1. Display selected subjects",0ah,0dh>
mwrite <"2. Enter marks for a subject",0ah,0dh>
```

```
mwrite <"3. View academic progress",0ah,0dh>
mwrite <"4. Generate Fee Challan",0ah,0dh>
mwrite <"5. Generate Transcript",0ah,0dh>;manage marks
mwrite <"6. Set timetable",0ah,0dh>
mwrite <"7. View timetable",0ah,0dh>
mwrite <"8. Logout",0ah,0dh>
ret
displaymenu endp

login proc,
var1:ptr byte
mwrite "Enter student ID to proceed: "
mov edx,var1
mov ecx,50
call readstring
push eax
push var1
call checkID
ret
login endp

checkID proc,
stra:ptr byte,
sz:dword
mov eax,sz
cmp eax,7
jne bad
```

```
mov esi,stra

mov bl,48
cmp [esi],bl
jb bad
cmp [esi+1],bl
jb bad


mov bl,57
cmp [esi],bl
ja bad
cmp [esi+1],bl
ja bad

mov bl,97
cmp [esi+2],bl
jb c1
mov bl,122
cmp [esi+2],bl
ja bad
jmp c2
c1:
mov bl,65
cmp [esi+2],bl
jb bad
mov bl,90
```

```
cmp [esi+2],bl
ja bad
c2:
add esi,3
mov ecx,4
l5:
mov al,48
cmp [esi],al
jb bad
mov al,57
cmp [esi],al
ja bad
inc esi
loop l5
mov valid,1
jmp e1
bad:
mov valid,0
e1:ret
checkID endp



settable proc uses esi eax ecx edx ebx edi,
ad1:ptr dword,
ad2:ptr dword
```

```
local
count1:dword,count2:dword,subcount[6]:dword,mcc1:dword,mcc2:dwo
rd
mov count1,0
mov count2,0
mov edx,offset days
mov esi,ad2
mov ebx,ad1
mov edi,0

mov ecx,5
mov mcc2,5
inloop:
mov ebx,ad1

push ecx
mov ecx,6
mov esi,0
tlp:
mov subcount[esi*type subcount],0
inc esi
loop tlp
pop ecx

call clrscr
mwrite <0ah,0dh,"Schedule for ">
call writestring
```

```
mwrite <": ",0ah,0dh,0ah,0dh>
call readchar

push ecx
mov ecx,8
mov mcc1,8
mov count1,0
mov count2,0
ilp2:

mwrite <0ah,0dh,"Choosing for slot ">
inc count1
mov eax,count1
call writedec
mwrite <": ",0ah,0dh>
call crlf

tr1:
mwrite <"1. Class",0ah,0dh,"2. Break",0ah,0dh>
mwrite <"Enter your choice: ">
call readint
cmp eax,2
je  break1

call displaysubs
call crlf
mwrite <"Enter your choice: ">
```

```
call readint
dec eax
mov esi,3
cmp subcount[eax*type subcount],esi
je ivd1
add subcount[eax*type subcount],1
mov [ebx+edi],eax
jmp cont1

break1:
mov eax,3
cmp count2,eax
je invldd1
inc count2
mov eax,6
mov [ebx+edi],eax ;break stored
jmp cont1

ivd1:
mwrite <"Subject already entered 3 times! Try again",0ah,0dh>
call crlf
call readchar
jmp tr1

invldd1:
mwrite <0ah,0dh,"Maximum number of breaks reached! Please try
again",0ah,0dh>
```

```
call crlf
call readchar
jmp tr1

cont1:
mwrite <0ah,0dh,"Entry stored successfully",0ah,0dh>
mov eax,250
call delay
add ebx,rowsize
sub mcc1,1
jnz ilp2
pop ecx

add edi,4
add edx,daysize
sub mcc2,1
jnz inloop

call crlf
call crlf
mwrite "TimeTable has been successfully set! Press any key to
continue."
call readchar
ret
settable endp

viewcurrent proc
```

```
local rowi:dword,coli:dword,sechalf:dword
mov sechalf,0

invoke GetLocalTime, ADDR localt
call clrscr
movzx eax,localt.wdayofweek
mov coli,eax
mov eax,5
cmp coli,eax
ja frday

movzx eax,localt.whour
cmp eax,16
ja badtime
cmp eax,8
jb badtime

sub eax,8
mov rowi,eax
mov ebx,offset arr1
mov eax,rowsize
mul rowi
add ebx,eax
sub coli,1
mov esi,coli
mov eax,[ebx+esi*type arr1]
cmp eax,6
```

```
jne fine1
mov eax,rowi
inc eax
mwrite <"Slot # ">
call writedec
mwrite <" ===>">
mwrite <0ah,0dh," FREE - YOU DO NOT HAVE ANY CLASSES AT THE
MOMENT",0ah,0dh>
call crlf
jmp scc4
fine1:
push eax
mov eax,rowi
inc eax
mwrite <"Slot # ">
call writedec
mwrite <" ===>">
pop eax
mov edi,eax
mov edx,mysubjects[edi*type mysubjects]
call writestring
call crlf
jmp scc4

badtime:
cmp eax,12
jna e1
```

```
mwrite <0ah,0dh,"It is past 04:00 PM. The university is now closed.",0ah,0dh>

jmp scc4

e1:

mwrite <0ah,0dh,"Too early for class! Classes don't start until 08:00 AM",0ah,0dh>

jmp scc4


frday:

mwrite <0ah,0dh,"Today is ">

mov eax,6

cmp coli,eax

je satday

mwrite <"Sunday">

jmp contin1

satday:

mwrite <"Saturday">

contin1:

mwrite <0ah,0dh,"No classes today.">

scc4:mwrite <0ah,0dh,"Press any key to continue...">

call readchar


ret

viewcurrent endp


viewtable proc uses eax edx ecx ebx esi edi

local co1:dword,co2:dword ,mc1:dword
```

```
mov co1,8
mov co2,9

mwrite <"Do you wish to view: ",0ah,0dh,"1. WHOLE
TIMETABLE",0ah,0dh,"2. CURRENT CLASS? ">
call crlf
call crlf
mwrite <"Enter choice: ">
call readint
cmp eax,2
jne proceedp
pushad
call viewcurrent
popad
jmp scc3

proceedp:
call clrscr
mov edx,offset days
mov ecx,5
mov esi,0

call crlf
mwrite <"| SLOTS ||  ">
tr2:
call writestring
mwrite <" | ">
```

```
add edx,daysize
loop tr2

call crlf
mov ecx,8
mov ebx,offset arr1
mov mc1,8
olp1:

mov eax,12
cmp co2,eax
jna chk1
mov co2,1
jmp chk1

chk1:
cmp co1,eax
jna prc1
mov co1,1

prc1:
mwrite <"| ">
mov eax,co1
call writedec
mwrite <" - ">
mov eax,co2
call writedec
```

```
mwrite "||"
inc co2
inc co1

mov esi,0
push ecx
mov ecx,5
olp2:
mov edi,[ebx+esi]
mov eax,6
cmp edi,eax
jb moveon
mwrite " FREE "
jmp cnt1
moveon:
mov edx,mysubjects[edi*type mysubjects]
push esi
mov esi,edx
mov edi,offset tempstr
push ecx
mov ecx,5
cld
rep movsd
pop ecx
pop esi
mov edx,offset tempstr
call writestring
```

```
cnt1:
mwrite ".    | "
add esi,4
loop olp2

pop ecx
add ebx,rowsize
call crlf
call crlf
sub mc1,1
jnz olp1
scc3:
call crlf
mwrite <0ah,0dh,"Press any key to continue..">
call readchar
ret
viewtable endp




End main
```

RESULTS/SOFTWARE SIMULATION AND DISCUSSION:

TESTING TIMETABLE AND LOCAL TIME FUNCTIONALITY:

- Displaying the generated timetable after user input
    - Input is taken for every day of the week

- For every day, there are 8 slots as shown below, so input taken 8 times per day
- Choice of either break or class is given
-

```
C:\Users\Ali Nadir\source\repos\Pro

Schedule for Monday    :

Choosing for slot 1:

1. Class
2. Break
Enter your choice: 2

Entry stored successfully

Choosing for slot 2:

1. Class
2. Break
Enter your choice:
```

-

- If class is selected, user enters choice of subject to be added to slot
- If break selected, index manipulated in such a way that table displays "FREE" on console
- For both breaks and a certain subject, a maximum of three breaks and maximum of three slots of the "SAME SUBJECT" are allowed
- **Same BREAK:**

```
Choosing for slot 1:

1. Class
2. Break
Enter your choice: 2          1

Entry stored successfully

Choosing for slot 2:

1. Class
2. Break
Enter your choice: 2          2

Entry stored successfully

Choosing for slot 3:

1. Class
2. Break
Enter your choice: 2
                              3
Entry stored successfully

Choosing for slot 4:

1. Class
2. Break
Enter your choice: 2

Maximum number of breaks reached! Please try again
```

```
2. Break
1. Sub 1
2. Sub 2
3. Sub 3
4. Sub 4
5. Sub 5
6. Sub 6


Enter your choice: 1
```

```
Choosing for slot 5:

1. Class
2. Break
Enter your choice: 1
1. Sub 1
2. Sub 2
3. Sub 3
4. Sub 4
5. Sub 5
6. Sub 6


Enter your choice: 1

Entry stored successfully
```

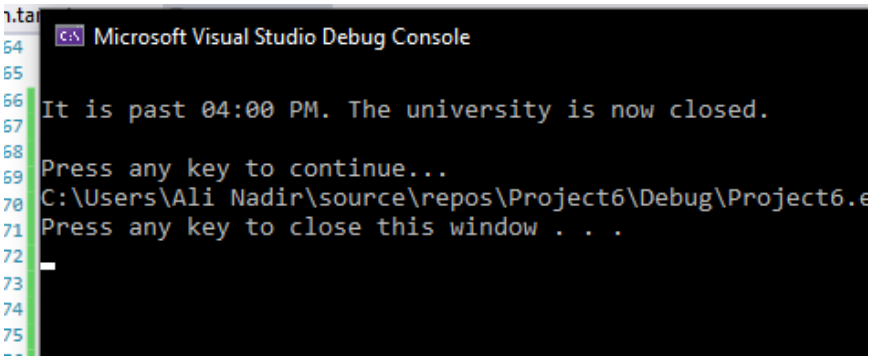
```
Choosing for slot 6:

1. Class
2. Break
Enter your choice: 1
1. Sub 1
2. Sub 2
3. Sub 3
4. Sub 4
5. Sub 5
6. Sub 6


Enter your choice: 1

Entry stored successfully
```

```
Choosing for slot 7:

1. Class
2. Break
Enter your choice: 1
1. Sub 1
2. Sub 2
3. Sub 3
4. Sub 4
5. Sub 5
6. Sub 6


Enter your choice: 1
Subject already entered 3 times! Try again
```

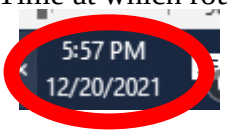
- **Output for WHOLE:**

```
Microsoft Visual Studio Debug Console

| SLOTS ||  Monday     |  Tuesday    |  Wednesday  |  Thursday   |  Friday     |
| 8 - 9|| FREE .   |  FREE .   |  FREE .   |  FREE .   | Sub 1.   |

| 9 - 10|| FREE .   |  FREE .   |  FREE .   |  FREE .   | Sub 1.   |

| 10 - 11|| FREE .   |  FREE .   |  FREE .   |  FREE .   | Sub 1.   |

| 11 - 12||Sub 1.   | Sub 1.   | Sub 1.   | Sub 1.   | Sub 2.   |

| 12 - 1||Sub 1.   | Sub 1.   | Sub 1.   | Sub 1.   | Sub 2.   |

| 1 - 2||Sub 1.   | Sub 1.   | Sub 1.   | Sub 1.   |  FREE .   |

| 2 - 3||Sub 2.   | Sub 2.   | Sub 2.   | Sub 2.   |  FREE .   |

| 3 - 4||Sub 2.   | Sub 2.   | Sub 2.   | Sub 2.   |  FREE .   |
```

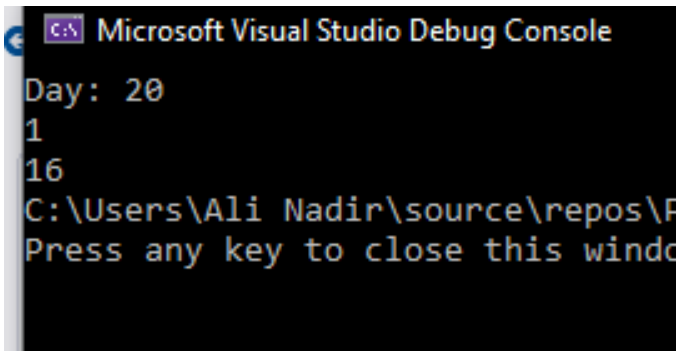
Output for CURRENT:



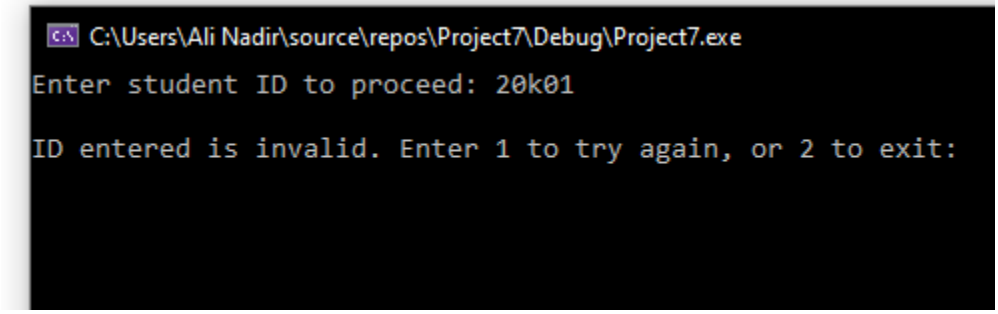
- Time at which routine was tested:

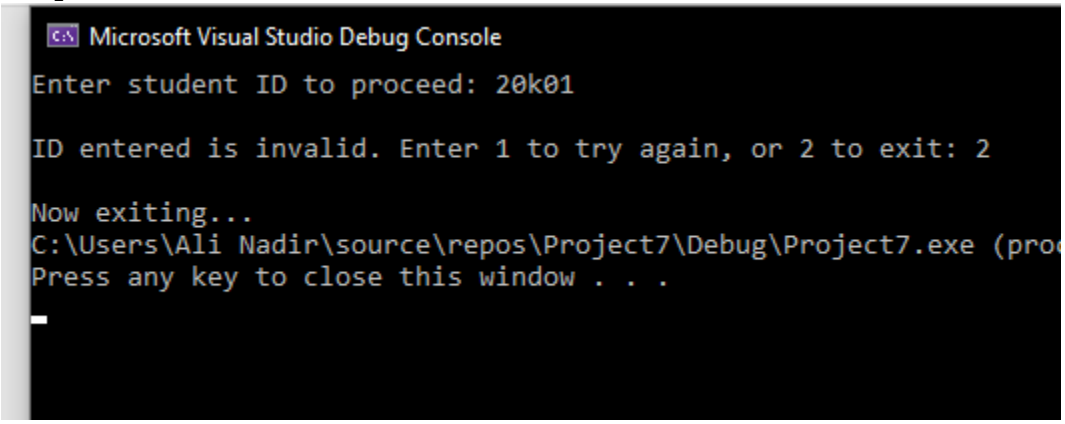


- **Checking if LPSYSTEMTIME structure from kernel32.lib library is working or not:**
  - First line tests the day number returned by member wDay of structure variable LOCALT
  - Second line shows the day of the week returned by attribute wDayofWeek (1-7) (Tested on Monday so displays 1)
  - Third line shows value of structure attribute wHour (Tested between 4PM-5PM so shows 16 since 16 represents 4th hour after noon (12+4=16)



TESTING STUDENT ID VALIDATION:

- **Invalid input:**
- 

- **Output:**
- 

- **Valid input:**

-
```
C:\Users\Ali Nadir\source\repos\Project7\Debug\Projec
Enter student ID to proceed: 20k0325
```
-
- **Ouput:**
```
C:\Users\Ali Nadir\source\repos\Project7\Debug\Projec
Enter student ID to proceed: 20k0325

Welcome user 20k0325
Please select which year you are in:
1. First
2. Second
3. Third
```
-

SUBJECT ENTRY:

- Entering 1 as year input
```
Welcome user 20k0325
Please select which year you are in:
1. First
2. Second
3. Third
4. Fourth
Enter choice: 1

Please select any 6 subjects:
1. English Composition and Comprehension
2. Applied Physics
3. Islamiyat
4. Programming Fundamentals
5. Calculus
6. Information and Computer Technology
7. Object Oriented Programming
8. Communication and Presentation Skills
9. Digital Logic Design
10. Pakistan Studies
11. Differential Equations

Enter your choice: _
```
-
- Selecting **6 different** subjects:

- For the same input **twice:**

```
Please select any 6 subjects:
1. English Composition and Comprehension
2. Applied Physics
3. Islamiyat
4. Programming Fundamentals
5. Calculus
6. Information and Computer Technology
7. Object Oriented Programming
8. Communication and Presentation Skills
9. Digital Logic Design
10. Pakistan Studies
11. Differential Equations

Enter your choice: 1

Enter your choice: 2

Enter your choice: 2
Invalid choice entered.
Enter your choice: _
```

```
Enter your choice: 2
Invalid choice entered.
Enter your choice: 3

Enter your choice: 4

Enter your choice: 5

Enter your choice: 6

Subjects successfully stored. You have chosen:
1. English Composition and Comprehension
2. Applied Physics
3. Islamiyat
4. Programming Fundamentals
5. Calculus
6. Information and Computer Technology

Enter any key to proceed...
```

- Selected subjects displayed, now pause console output with **readchar**

MARKS ENTRY:

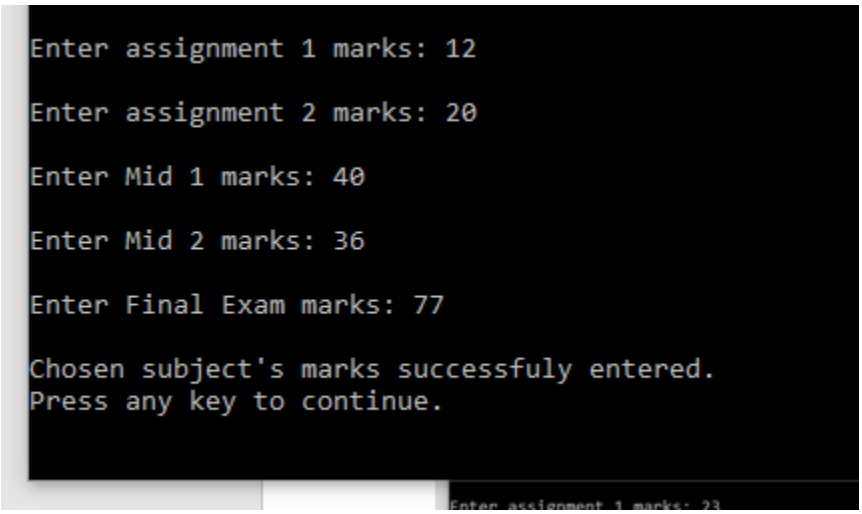
- If transcript not yet generated, allows marks entry
- Else:

```
Please select one of the following:
1. Display selected subjects
2. Enter marks for a subject
3. View academic progress
4. Generate Fee Challan
5. Generate Transcript
6. Set timetable
7. View timetable
8. Logout
2

Marks have now been locked in the system. Can not proceed
Press any key to continue.
```

- Selecting subject no. 2 for entry:
- Entering invalid marks (Assignments are marked out of 20)

```
Select subject whose marks are to be entered:
1. English Composition and Comprehension
2. Applied Physics
3. Islamiyat
4. Programming Fundamentals
5. Calculus
6. Information and Computer Technology

Enter choice: 2

Enter assignment 1 marks: 23

Incorrect entry. Please enter marks again.

Enter assignment 1 marks: _
```

- Prompted for correct entry

- 

- Redirect to menu after marks have been successfully entered

- If user chooses to reenter marks for an already recorded subject:
  - Ask for confirmation to overwrite marks
  -

-
```
Select subject whose marks are to be entered:
1. English Composition and Comprehension
2. Applied Physics
3. Islamiyat
4. Programming Fundamentals
5. Calculus
6. Information and Computer Technology

Enter choice: 2

Chosen subject's marks have already been entered.
Do you wish to overwrite subject marks? Enter 1 if yes, else for no:
```

- Invalid Mid 1/2 marks (mustn't be greater than 40)
-
```
Enter Mid 2 marks: 42

Incorrect entry. Please enter marks again.
```

- Invalid final exam marks (mustn't be greater than 100)

```
Enter Mid 1 marks: 35

Enter Mid 2 marks: 37

Enter Final Exam marks: 101

Incorrect entry. Please enter marks again.
```

VIEWING RECORDS:

- Viewing record for recorded subject: (VALID)

```
Select subject whose record you wish to view:
1. English Composition and Comprehension
2. Applied Physics
3. Islamiyat
4. Programming Fundamentals
5. Calculus
6. Information and Computer Technology

Enter choice: 1
```

-

```
|| Assignment 1 | Assignment 2 |     Mid 1     |     Mid 2     |     Final     ||
||     15        |     15       |     35        |     37        |     88        ||
Press any key to view weightage breakdown.
|| Assignment 1 | Assignment 2 |     Mid 1     |     Mid 2     |     Final     ||
```

-

```
|| Assignment 1 | Assignment 2 |     Mid 1     |     Mid 2     |     Final     ||
||     15        |     15       |     35        |     37        |     88        ||
Press any key to view weightage breakdown.
|| Assignment 1 | Assignment 2 |     Mid 1     |     Mid 2     |     Final     ||
||     7.0 %     |     7.0 %    |     13.0 %    |     13.0 %    |     44.0 %
Grand Total for subject English Composition and Comprehension   is: 84
Overall grade:  A
Subject GPA: 4
Press any key to continue...
```

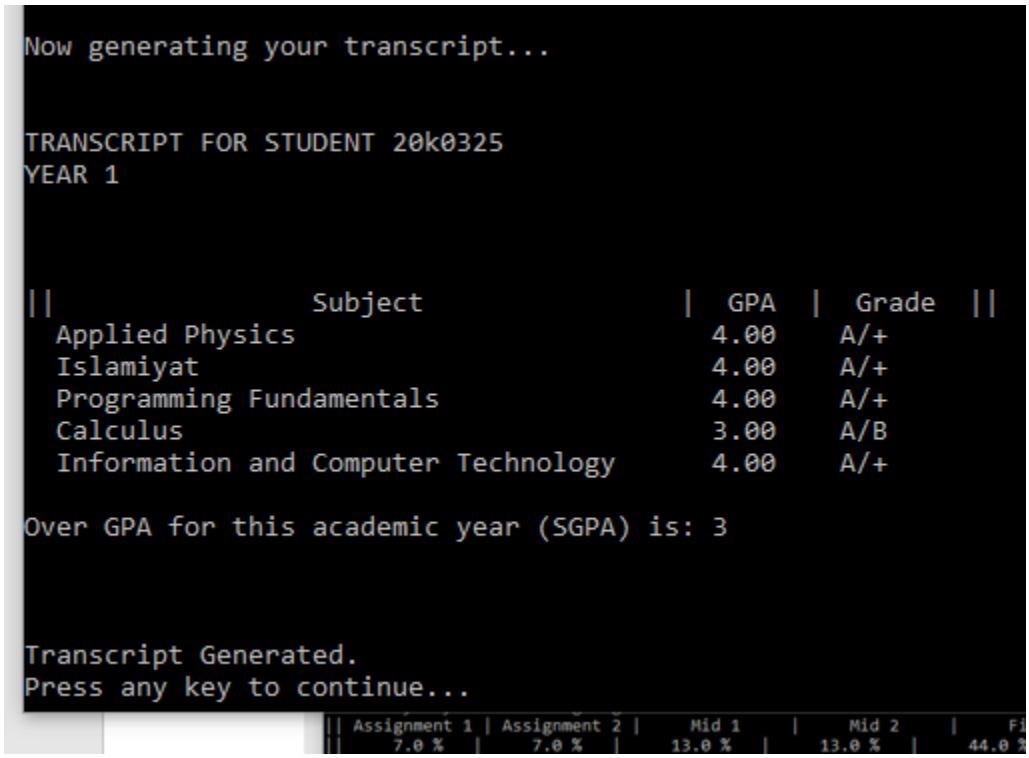
-

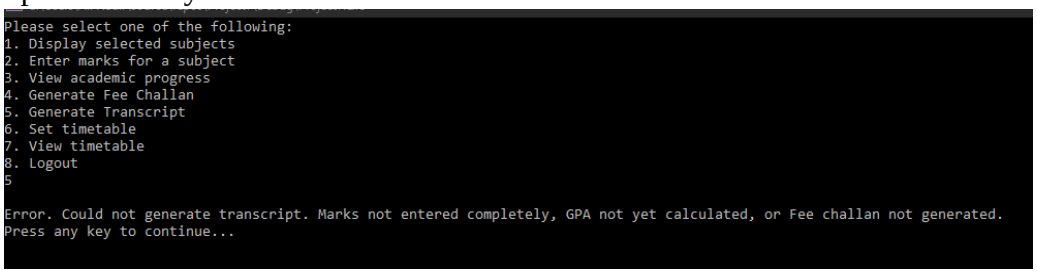
- Invalid choice (subject not recorded yet)

-
```
Select subject whose record you wish to view:
1. English Composition and Comprehension
2. Applied Physics
3. Islamiyat
4. Programming Fundamentals
5. Calculus
6. Information and Computer Technology

Enter choice: 5

Error: Marks have not been completely entered. Try again later.

Press any key to continue...
```

GENERATING TRANSCRIPT:

- Valid output:
  - Ask user for confirmation to lock in grades and proceed with generation
  -
```
Warning: Generating transcript will lock your
grades in the system. Enter 1 to continue, any
 other key to return:
```

- 

- Output upon invalidity:
  - 

  - Requires:
    - Marks to be entered completely
    - GPA to be generated for all subjects (by viewing records at least once)
    - Fee Challan to be generated

GENERATING FEE CHALLAN:

- Prompting user to enter credit hours for each subject:
  -
```
Enter credit hours for English Composition and Comprehension  : 3

Enter credit hours for Applied Physics                        : 3

Enter credit hours for Islamiyat                              : 2

Enter credit hours for Programming Fundamentals               : 2

Enter credit hours for Calculus                               : 3

Enter credit hours for Information and Computer Technology    : 3
```
- Generated fee challan:
  -
```
**********************************FEE CHALLAN FOR USER 20k0325**********************************


For English Composition and Comprehension  :
Credit Hours: 3
Cost per Credit Hour: PKR 5000

For Applied Physics                        :
Credit Hours: 3
Cost per Credit Hour: PKR 5000

For Islamiyat                              :
Credit Hours: 2
Cost per Credit Hour: PKR 5000

For Programming Fundamentals               :
Credit Hours: 2
Cost per Credit Hour: PKR 5000

For Calculus                               :
Credit Hours: 3
Cost per Credit Hour: PKR 5000

For Information and Computer Technology    :
Credit Hours: 3
Cost per Credit Hour: PKR 5000
******************Total charges due for the 6 chosen subjects: PKR 80000******************
*****************************************************************************************


Press any key to continue...
```
  -

- **Total displayed is PKR 80000** for the term
- **Cost for each year:**
  - 1st PKR 5000
  - 2nd PKR 7500
  - 3rd PKR 10000

CONCLUSION, COST, AND FUTURE WORK:

The solution solves the problem of computational complexity in that it solely uses runtime stack and memory for the storage, extraction and manipulation of record data without straining secondary storage with this data, as filing has not been used. File processing will cause unnecessarily slow read and write speeds, thus defeating the ultimate purpose of the application. Since users can readily view and pull the records they want to view individually instead of sifting through endless piles of multiple records of any number of subjects, the user friendliness and efficiency of the code can be asserted as well, as tabular data was observed to be less resourceful as individual record data of choosing. Also, the code relies heavily on the use of the runtime stack to manage memory as efficiently as possible throughout runtime. In the future, the system should be improved to allow multiple academic terms to be handled with the same efficiency as described before. GUI interface implementation should be focused on in the future as well to improve the UI/UX aspects of the application as well. Also, implementation of a purely structure based approach should be practiced in the long term for maximum efficiency.

REFERENCES:

What is an LMS? Definition, Features, and Use Cases (ispringsolutions.com)

Full article: Learning management systems: a review of the research methodology literature in Australia and China (tandfonline.com)

Full article: Learning management systems: a review of the research methodology literature in Australia and China (tandfonline.com)

visual studio - Include Files for MASM - Stack Overflow

Download The MASM32 SDK