include Irvine32.inc

; function prototypes

createEmployee proto, emp\_arr:PTR Employee, num\_emp:PTR DWORD

removeEmployee proto, emp\_arr:PTR Employee, num\_emp: ptr DWORD

modifyEmployee proto, emp\_arr:PTR Employee, num\_emp:PTR DWORD

getEmployeeInfo proto, emp\_arr:PTR Employee, num\_emp:PTR DWORD

printAllEmployees proto, emp\_arr:PTR Employee, num\_emp:PTR DWORD

Employee struct

emp\_name byte 20 dup(?),0

emp\_id byte 4 dup(?),0

emp\_ph byte 15 dup(?),0

emp\_homeaddr byte 40 dup(?),0

emp\_email byte 20 dup(?),0

emp\_designation byte 20 dup(?),0

emp\_department byte 20 dup(?),0

emp\_work\_nature byte 20 dup(?),0 ;remote/on-site/hybrid

emp\_status byte 20 dup(?),0 ;hired/fired/notice-period/suspended

emp\_contract\_years dword 0

emp\_salary dword 0

Employee ends

.data

.stack 4096

max\_capacity = 50

;Array of employee structs

num\_employees dword 0

employee\_array Employee max\_capacity dup(<,,,,,,,,,,-1>)

;all the strings to be displayed

;menu options

menu\_str byte "MAIN MENU: ",0

space byte " ",0

s1 byte "ENTER CORRESPONDING KEY TO PROCEED TO SPECIFIC TASK : ",0

addemp\_str byte "[1] ADD AN EMPLOYEE",0

addingemp byte "Creating a new Employee entry:",0

rmvemp\_str byte "[2] REMOVE AN EMPLOYEE",0

emprmvId\_str byte "Enter the Employee ID of the Employee you want to delete",0

empmodify\_str byte "[3] EDIT EMPLOYEE INFORMATION",0

empmodifyId\_str byte "Enter the Employee ID of the Employee who's information you want to modify",0

empmodifyWhat\_str byte "What information would you like to modify for Employee ID: ",0

empinfo\_str byte "[4] VIEW EMPLOYEE INFORMATION ",0

empinfoId\_str byte "Enter the Employee ID of the Employee who's information you want to retrieve",0

empinfoall\_str byte "Information of Employee ID: ",0

allemp\_str byte "[5] DISPLAY ALL EMPLOYEES",0

listingall byte "List of all the information of each registered Employee:",0

nothinglist byte "There are no Registered Employees",0

printempnum\_str byte "[6] DISPLAY TOTAL REGISTERED EMPLOYEES",0

printempnum byte "Number of Employees currently registered are: ",0

exitsave\_str byte "[0] SAVE CHANGES AND EXIT",0

exitnosave\_str byte "[7] DISCARD CHANGES AND EXIT",0

empremoved byte "Employee removed successfully",0

empadded byte "Employee added successfully",0

empmodif byte "Employee modified successfully",0

chngsaved\_str byte "Changes saved successfully",0

chngnotsaved\_str byte "Changes have not been made",0

lowmaxcap\_str byte "Insufficient max capacity, minimum max capacity required is : ",0

maxcapreached byte "Sorry Max number of employees reached",0

empnotfound byte "Employee not found",0

invalid\_choice byte "Invalid choice, try again",0

invalsal\_str byte "Invalid Salary amount, please try again",0

invalcontract\_str byte "Invalid Contract years, please try again",0

inval\_id\_str byte "Invalid ID, must be Unique and exactly 4 digits long. Please try again",0

notold\_id\_str byte "New ID cannot be same as the previous one",0

;modify options strings for employee attributes

allinfo\_str byte "0: Employee's Name", 0dh, 0ah

byte "1: Employee's ID", 0dh, 0ah

byte "2: Employee's Phone Number", 0dh, 0ah

byte "3: Employee's Home Address", 0dh, 0ah

byte "4: Employee's Email Address", 0dh, 0ah

byte "5: Employee's Designation", 0dh, 0ah

byte "6: Employee's Department", 0dh, 0ah

byte "7: Employee's Nature of Work", 0dh, 0ah

byte "8: Employee's Employment Status", 0dh, 0ah

byte "9: Employee's Contract Years", 0dh, 0ah

byte "10: Employee's Salary", 0dh, 0ah,0

;info entering request strings

entername\_str byte "Enter Employee's Name", 0dh, 0ah,0

enterid\_str byte "Enter Employee's ID, must be exactly 4 digits", 0dh, 0ah,0

enterph\_str byte "Enter Employee's Phone Number", 0dh, 0ah,0

enteraddr\_str byte "Enter Employee's Home Address", 0dh, 0ah,0

enteremail\_str byte "Enter Employee's Email Address", 0dh, 0ah,0

enterdesig\_str byte "Enter Employee's Designation", 0dh, 0ah,0

enterdepart\_str byte "Enter Employee's Department", 0dh, 0ah,0

enternature\_str byte "Enter Employee's Nature of Work", 0dh, 0ah,0

enterstatus\_str byte "Enter Employee's Employment Status", 0dh, 0ah,0

entercontract\_str byte "Enter Employee's Contract Years", 0dh, 0ah,0

entersalary\_str byte "Enter Employee's Salary", 0dh, 0ah,0

;info labelling

nametext byte "Employee's Name: ", 0

id byte "Employee's ID: ", 0

phone byte "Employee's Phone Number: ",0

address byte "Employee's Home Address: ", 0

email byte "Employee's Email Address: ", 0

desig byte "Employee's Designation: ", 0

depart byte "Employee's Department: ", 0

nature byte "Employee's Nature of Work: ", 0

status byte "Employee's Employment Status: ", 0

contract byte "Employee's Contract Years: ",0

salary byte "Employee's Salary: ",0

s2 byte "ENTER YOUR INPUT HERE -> ",0

;file handle for filing

filehandle handle ?

;file name

filename byte "employeeDB",0

caption byte "SAVE AND EXIT",0

question byte "ARE YOU SURE YOU WANT TO SAVE CHANGES BEFORE EXITING? ",0

caption1 byte "EXIT WITHOUT SAVING",0

question1 byte "ARE YOU SURE YOU WANT EXIT WITHOUT SAVING CHANGES? ",0

.code

main proc

invoke CreateFile, ADDR fileName, GENERIC\_READ or GENERIC\_WRITE, 0, NULL, OPEN\_EXISTING, FILE\_ATTRIBUTE\_NORMAL, NULL

mov filehandle, eax

cmp eax, INVALID\_HANDLE\_VALUE

je mainloop

invoke ReadFile, filehandle, ADDR num\_employees, type num\_employees, NULL, NULL

mov ecx, sizeof Employee

mov eax, num\_employees

inc eax

mul ecx

invoke ReadFile, filehandle, offset num\_employees + type num\_employees, eax, NULL, NULL

invoke CloseHandle, filehandle

mov ebx, max\_capacity

cmp ebx, num\_employees

jg mainloop

mov edx, offset lowmaxcap\_str

call writestring

mov eax, num\_employees

inc eax

call writedec

call crlf

jmp skip2

mainloop:

mov eax, (white\*16)+(black)

call settextcolor

call clrscr

lea edx, space

call crlf

mov ecx, 52

print\_spacex:

call writestring

loop print\_spacex

mov edx, offset menu\_str

call writestring

call crlf

call crlf

call crlf

call crlf

lea edx, s1

call writestring

call crlf

call crlf

call crlf

lea edx, space

call crlf

mov ecx, 35

print\_space0:

call writestring

loop print\_space0

mov eax, "-"

mov ecx, 50

print\_dash:

call writechar

Loop print\_dash

call crlf

lea edx, space

call crlf

mov ecx, 40

print\_space1:

call writestring

loop print\_space1

mov edx, offset exitsave\_str

call writestring

call crlf

lea edx, space

call crlf

mov ecx, 40

print\_space2:

call writestring

loop print\_space2

mov edx, offset addemp\_str

call writestring

call crlf

lea edx, space

call crlf

mov ecx, 40

print\_space3:

call writestring

loop print\_space3

mov edx, offset rmvemp\_str

call writestring

call crlf

lea edx, space

call crlf

mov ecx, 40

print\_space4:

call writestring

loop print\_space4

mov edx, offset empmodify\_str

call writestring

call crlf

lea edx, space

call crlf

mov ecx, 40

print\_space5:

call writestring

loop print\_space5

mov edx, offset empinfo\_str

call writestring

call crlf

lea edx, space

call crlf

mov ecx, 40

print\_space6:

call writestring

loop print\_space6

mov edx, offset allemp\_str

call writestring

call crlf

lea edx, space

call crlf

mov ecx, 40

print\_space7:

call writestring

loop print\_space7

mov edx, offset printempnum\_str

call writestring

call crlf

lea edx, space

call crlf

mov ecx, 40

print\_space8:

call writestring

loop print\_space8

mov edx, offset exitnosave\_str

call writestring

call crlf

;call crlf

lea edx, space

call crlf

mov ecx, 35

print\_space00:

call writestring

loop print\_space00

mov eax, "-"

mov ecx, 50

print\_dash1:

call writechar

Loop print\_dash1

call crlf

lea edx, s2

call writestring

call readdec

cmp eax, 0

je exit\_save

cmp eax, 7

je exit\_nosave

cmp eax, 1

je add\_emp

cmp eax, 2

je rmv\_emp

cmp eax, 3

je modif

cmp eax, 4

je getinfo

cmp eax, 5

je allemp

cmp eax, 6

je getnumemp

jmp inval

inter1:

jmp mainloop

add\_emp:

call clrscr

invoke createEmployee, addr employee\_array, offset num\_employees

call WaitMsg

call clrscr

jmp skip

rmv\_emp:

call clrscr

invoke removeEmployee, addr employee\_array, addr num\_employees

call WaitMsg

call clrscr

jmp skip

modif:

call clrscr

invoke modifyEmployee, addr employee\_array, addr num\_employees

call WaitMsg

call clrscr

jmp skip

getinfo:

call clrscr

invoke getEmployeeInfo, addr employee\_array, addr num\_employees

call WaitMsg

call clrscr

jmp skip

allemp:

call clrscr

invoke printAllEmployees, addr employee\_array, addr num\_employees

call WaitMsg

call clrscr

jmp skip

getnumemp:

call clrscr

mov edx, offset printempnum

call writestring

mov eax, num\_employees

call writedec

call crlf

call crlf

call WaitMsg

call clrscr

jmp skip

inval:

call clrscr

mov edx, offset invalid\_choice

call writestring

call crlf

call crlf

call WaitMsg

call clrscr

jmp skip

skip:

jmp inter1 ;main loop ending

exit\_save:

mov ebx, OFFSET caption

mov edx, OFFSET question

call MsgBoxAsk

cmp eax, 6

jne jumppp

invoke CreateFile, ADDR fileName, GENERIC\_READ or GENERIC\_WRITE, 0, NULL, OPEN\_ALWAYS, FILE\_ATTRIBUTE\_NORMAL, NULL

mov filehandle, eax

mov ecx, num\_employees

inc ecx

mov eax, sizeof Employee

mul ecx

add eax, type num\_employees

invoke WriteFile, filehandle, ADDR num\_employees, eax, NULL, NULL

invoke CloseHandle, filehandle

call crlf

mov edx, offset chngsaved\_str

call writestring

call crlf

jmp skip2

exit\_nosave:

mov ebx, OFFSET caption1

mov edx, OFFSET question1

call MsgBoxAsk

cmp eax, 6

jne mainloop

jumppp:

call crlf

mov edx, offset chngnotsaved\_str

call writestring

call crlf

skip2:

exit

main endp

; function definitions

createEmployee proc uses ecx, emp\_arr:PTR Employee, num\_emp:PTR DWORD

local helper\_str[30]:byte

mov esi, num\_emp

mov ecx, [esi]

cmp ecx, max\_capacity

je maxed

inc ecx

mov esi, emp\_arr

l1:

mov eax, (Employee ptr [esi]).emp\_salary

cmp eax, -1

je found

add esi, sizeof employee

loop l1

maxed:

mov edx, offset maxcapreached

call writestring

call crlf

ret

found:

mov edx, offset addingemp

call writestring

call crlf

call crlf

mov edx, offset entername\_str

call writestring

call crlf

lea edx, (Employee ptr [esi]).emp\_name

mov ecx, sizeof (Employee ptr [esi]).emp\_name

call readstring

id\_again:

call crlf

mov edx, offset enterid\_str

call writestring

call crlf

lea edx, helper\_str

mov ecx, sizeof helper\_str

call readstring

call crlf

cmp eax, 4

jne inval\_id

mov edi, num\_emp

mov ecx, [edi]

inc ecx

mov edi, emp\_arr

checkuniq:

mov eax, (Employee ptr [edi]).emp\_salary

cmp eax, -1

je next

lea edx, helper\_str

INVOKE Str\_compare, edx, ADDR (Employee ptr [edi]).emp\_id

jne next

jmp inval\_id

next:

add edi, sizeof employee

loop checkuniq

jmp ok\_id

inval\_id:

mov edx, offset inval\_id\_str

call writestring

call crlf

jmp id\_again

ok\_id:

lea edx, helper\_str

INVOKE Str\_copy, edx, ADDR (Employee ptr [esi]).emp\_id

mov edx, offset enterph\_str

call writestring

call crlf

lea edx, (Employee ptr [esi]).emp\_ph

mov ecx, sizeof (Employee ptr [esi]).emp\_ph

call readstring

call crlf

mov edx, offset enteraddr\_str

call writestring

call crlf

lea edx, (Employee ptr [esi]).emp\_homeaddr

mov ecx, sizeof (Employee ptr [esi]).emp\_homeaddr

call readstring

call crlf

mov edx, offset enteremail\_str

call writestring

call crlf

lea edx, (Employee ptr [esi]).emp\_email

mov ecx, sizeof (Employee ptr [esi]).emp\_email

call readstring

call crlf

mov edx, offset enterdesig\_str

call writestring

call crlf

lea edx, (Employee ptr [esi]).emp\_designation

mov ecx, sizeof (Employee ptr [esi]).emp\_designation

call readstring

call crlf

mov edx, offset enterdepart\_str

call writestring

call crlf

lea edx, (Employee ptr [esi]).emp\_department

mov ecx, sizeof (Employee ptr [esi]).emp\_department

call readstring

call crlf

mov edx, offset enternature\_str

call writestring

call crlf

lea edx, (Employee ptr [esi]).emp\_work\_nature

mov ecx, sizeof (Employee ptr [esi]).emp\_work\_nature

call readstring

call crlf

mov edx, offset enterstatus\_str

call writestring

call crlf

lea edx, (Employee ptr [esi]).emp\_status

mov ecx, sizeof (Employee ptr [esi]).emp\_status

call readstring

invalcontract:

call crlf

mov edx, offset entercontract\_str

call writestring

call crlf

call readint

cmp eax, -1

jg contractok

mov edx, offset invalcontract\_str

call writestring

call crlf

jmp invalcontract

contractok:

mov (Employee ptr [esi]).emp\_contract\_years, eax

invalsal:

call crlf

mov edx, offset entersalary\_str

call writestring

call crlf

call readint

cmp eax, -1

jg salaryok

mov edx, offset invalsal\_str

call writestring

call crlf

jmp invalsal

salaryok:

mov (Employee ptr [esi]).emp\_salary, eax

mov edx, offset empadded

call writestring

call crlf

mov esi, num\_emp

mov edx, [esi]

inc edx

mov [esi], edx

ret

createEmployee endp

removeEmployee proc uses ecx, emp\_arr:PTR Employee, num\_emp: ptr DWORD

local helper\_str[5]:byte

mov helper\_str[4], 0 ;to account for null termination

mov edx, offset emprmvId\_str

call writestring

call crlf

lea edx, helper\_str

mov ecx, sizeof helper\_str

call readstring

mov esi, num\_emp

mov ecx, [esi]

inc ecx

mov esi, emp\_arr

l2:

lea edx, helper\_str

INVOKE Str\_compare, edx , ADDR (Employee ptr [esi]).emp\_id

jne next

mov eax, (Employee ptr [esi]).emp\_salary

cmp eax, -1

je next

jmp found2

next:

add esi, sizeof employee

loop l2

mov edx, offset empnotfound

call writestring

call crlf

ret

found2:

mov (Employee ptr [esi]).emp\_salary, -1

mov esi, num\_emp

mov edx, [esi]

dec edx

mov [esi], edx

mov edx, offset empremoved

call writestring

call crlf

ret

removeEmployee endp

modifyEmployee proc uses ecx, emp\_arr:PTR Employee, num\_emp:PTR DWORD

local helper\_str[5]:byte

mov helper\_str[4], 0 ;to account for null termination

mov edx, offset empmodifyId\_str

call writestring

call crlf

lea edx, helper\_str

mov ecx, sizeof helper\_str

call readstring

mov esi, num\_emp

mov ecx, [esi]

inc ecx

mov esi, emp\_arr

l3:

lea edx, helper\_str

INVOKE Str\_compare, edx , ADDR (Employee ptr [esi]).emp\_id

jne next

mov eax, (Employee ptr [esi]).emp\_salary

cmp eax, -1

je next

jmp found3

next:

add esi, sizeof employee

loop l3

mov edx, offset empnotfound

call writestring

call crlf

ret

found3:

modifagain:

call crlf

mov edx, offset empmodifyWhat\_str

call writestring

lea edx, (Employee ptr [esi]).emp\_id

call writestring

call crlf

call crlf

mov edx, offset allinfo\_str

call writestring

call crlf

call readdec

call crlf

cmp eax, 0

je namemod

cmp eax, 1

je idmod

cmp eax, 2

je phmod

cmp eax, 3

je addrmod

cmp eax, 4

je emailmod

cmp eax, 5

je desigmod

cmp eax, 6

je departmod

cmp eax, 7

je naturemod

cmp eax, 8

je statusmod

cmp eax, 9

je contractmod

cmp eax, 10

je salmod

jmp invalmod

namemod:

mov edx, offset entername\_str

call writestring

call crlf

lea edx, (Employee ptr [esi]).emp\_name

mov ecx, sizeof (Employee ptr [esi]).emp\_name

call readstring

jmp nomod

idmod:

id\_again:

call crlf

mov edx, offset enterid\_str

call writestring

call crlf

lea edx, helper\_str

mov ecx, sizeof helper\_str

call readstring

call crlf

cmp eax, 4

jne inval\_id

lea edx, helper\_str

INVOKE Str\_compare, edx, ADDR (Employee ptr [esi]).emp\_id

je notold\_id

mov edi, num\_emp

mov ecx, [edi]

inc ecx

mov edi, emp\_arr

checkuniq:

mov eax, (Employee ptr [edi]).emp\_salary

cmp eax, -1

je next2

lea edx, helper\_str

INVOKE Str\_compare, edx, ADDR (Employee ptr [edi]).emp\_id

jne next2

jmp inval\_id

next2:

add edi, sizeof employee

loop checkuniq

jmp ok\_id

notold\_id:

mov edx, offset notold\_id\_str

call writestring

call crlf

jmp id\_again

inval\_id:

mov edx, offset inval\_id\_str

call writestring

call crlf

jmp id\_again

ok\_id:

lea edx, helper\_str

INVOKE Str\_copy, edx, ADDR (Employee ptr [esi]).emp\_id

jmp nomod

phmod:

mov edx, offset enterph\_str

call writestring

call crlf

lea edx, (Employee ptr [esi]).emp\_ph

mov ecx, sizeof (Employee ptr [esi]).emp\_ph

call readstring

jmp nomod

addrmod:

mov edx, offset enteraddr\_str

call writestring

call crlf

lea edx, (Employee ptr [esi]).emp\_homeaddr

mov ecx, sizeof (Employee ptr [esi]).emp\_homeaddr

call readstring

jmp nomod

emailmod:

mov edx, offset enteremail\_str

call writestring

call crlf

lea edx, (Employee ptr [esi]).emp\_email

mov ecx, sizeof (Employee ptr [esi]).emp\_email

call readstring

jmp nomod

desigmod:

mov edx, offset enterdesig\_str

call writestring

call crlf

lea edx, (Employee ptr [esi]).emp\_designation

mov ecx, sizeof (Employee ptr [esi]).emp\_designation

call readstring

jmp nomod

departmod:

mov edx, offset enterdepart\_str

call writestring

call crlf

lea edx, (Employee ptr [esi]).emp\_department

mov ecx, sizeof (Employee ptr [esi]).emp\_department

call readstring

jmp nomod

naturemod:

mov edx, offset enternature\_str

call writestring

call crlf

lea edx, (Employee ptr [esi]).emp\_work\_nature

mov ecx, sizeof (Employee ptr [esi]).emp\_work\_nature

call readstring

jmp nomod

statusmod:

mov edx, offset enterstatus\_str

call writestring

call crlf

lea edx, (Employee ptr [esi]).emp\_status

mov ecx, sizeof (Employee ptr [esi]).emp\_status

call readstring

jmp nomod

contractmod:

invalcontract:

mov edx, offset entercontract\_str

call writestring

call crlf

call readint

cmp eax, -1

jg contractok

mov edx, offset invalcontract\_str

call writestring

call crlf

jmp invalcontract

contractok:

mov (Employee ptr [esi]).emp\_contract\_years, eax

jmp nomod

salmod:

invalsal:

mov edx, offset entersalary\_str

call writestring

call crlf

call readint

cmp eax, -1

jg salaryok

mov edx, offset invalsal\_str

call writestring

call crlf

jmp invalsal

salaryok:

mov (Employee ptr [esi]).emp\_salary, eax

jmp nomod

invalmod:

mov edx, offset invalid\_choice

call writestring

call crlf

call crlf

jmp modifagain

nomod:

mov edx, offset empmodif

call writestring

call crlf

ret

modifyEmployee endp

getEmployeeInfo proc uses ecx, emp\_arr:PTR Employee, num\_emp:PTR DWORD

local helper\_str[5]:byte

mov helper\_str[4], 0 ;to account for null termination

mov edx, offset empinfoId\_str

call writestring

call crlf

lea edx, helper\_str

mov ecx, sizeof helper\_str

call readstring

mov esi, num\_emp

mov ecx, [esi]

inc ecx

mov esi, emp\_arr

l4:

lea edx, helper\_str

INVOKE Str\_compare, edx , ADDR (Employee ptr [esi]).emp\_id

jne next

mov eax, (Employee ptr [esi]).emp\_salary

cmp eax, -1

je next

jmp found4

next:

add esi, sizeof employee

loop l4

mov edx, offset empnotfound

call writestring

call crlf

ret

found4:

call crlf

mov edx, offset empinfoall\_str

call writestring

lea edx, (Employee ptr [esi]).emp\_id

call writestring

call crlf

call crlf

mov edx, offset nametext

call writestring

lea edx, (Employee ptr [esi]).emp\_name

call writestring

call crlf

mov edx, offset id

call writestring

lea edx, (Employee ptr [esi]).emp\_id

call writestring

call crlf

mov edx, offset phone

call writestring

lea edx, (Employee ptr [esi]).emp\_ph

call writestring

call crlf

mov edx, offset address

call writestring

lea edx, (Employee ptr [esi]).emp\_homeaddr

call writestring

call crlf

mov edx, offset email

call writestring

lea edx, (Employee ptr [esi]).emp\_email

call writestring

call crlf

mov edx, offset desig

call writestring

lea edx, (Employee ptr [esi]).emp\_designation

call writestring

call crlf

mov edx, offset depart

call writestring

lea edx, (Employee ptr [esi]).emp\_department

call writestring

call crlf

mov edx, offset nature

call writestring

lea edx, (Employee ptr [esi]).emp\_work\_nature

call writestring

call crlf

mov edx, offset status

call writestring

lea edx, (Employee ptr [esi]).emp\_status

call writestring

call crlf

mov edx, offset contract

call writestring

mov eax, (Employee ptr [esi]).emp\_contract\_years

call writedec

call crlf

mov edx, offset salary

call writestring

mov eax, (Employee ptr [esi]).emp\_salary

call writedec

call crlf

call crlf

ret

getEmployeeInfo endp

printAllEmployees proc uses ecx, emp\_arr:PTR Employee, num\_emp:PTR DWORD

mov esi, num\_emp

mov ecx, [esi]

inc ecx

mov esi, emp\_arr

mov eax, [esi]

cmp eax,0

je none

mov edx, offset listingall

call writestring

call crlf

l5:

mov eax, (Employee ptr [esi]).emp\_salary

cmp eax, -1

je next

call crlf

call crlf

mov edx, offset empinfoall\_str

call writestring

lea edx, (Employee ptr [esi]).emp\_id

call writestring

call crlf

call crlf

mov edx, offset nametext

call writestring

lea edx, (Employee ptr [esi]).emp\_name

call writestring

call crlf

mov edx, offset id

call writestring

lea edx, (Employee ptr [esi]).emp\_id

call writestring

call crlf

mov edx, offset phone

call writestring

lea edx, (Employee ptr [esi]).emp\_ph

call writestring

call crlf

mov edx, offset address

call writestring

lea edx, (Employee ptr [esi]).emp\_homeaddr

call writestring

call crlf

mov edx, offset email

call writestring

lea edx, (Employee ptr [esi]).emp\_email

call writestring

call crlf

mov edx, offset desig

call writestring

lea edx, (Employee ptr [esi]).emp\_designation

call writestring

call crlf

mov edx, offset depart

call writestring

lea edx, (Employee ptr [esi]).emp\_department

call writestring

call crlf

jmp skipinterloop

interloop:

jmp l5

skipinterloop:

mov edx, offset nature

call writestring

lea edx, (Employee ptr [esi]).emp\_work\_nature

call writestring

call crlf

mov edx, offset status

call writestring

lea edx, (Employee ptr [esi]).emp\_status

call writestring

call crlf

mov edx, offset contract

call writestring

mov eax, (Employee ptr [esi]).emp\_contract\_years

call writedec

call crlf

mov edx, offset salary

call writestring

mov eax, (Employee ptr [esi]).emp\_salary

call writedec

call crlf

call crlf

next:

add esi, sizeof employee

loop interloop

jmp skip3

none:

mov edx, offset nothinglist

call writestring

lea edx, (Employee ptr [esi]).emp\_status

call writestring

call crlf

skip3:

ret

printAllEmployees endp

end main