include irvine32.inc

;;;;;;SINGLY LINKED LIST ;;;;;;;

;;;FUNCTIONS INCLUDED;;;

;;1)Insert From Front

;;2)Delete From Front

;;3)Delete From Last

;;4)Insert From Last

;;5)Insert At Position

;;6)Delete From Position

;;7)Print All Nodes

;;8)Delete All Nodes

.data

insertnode proto , a:dword

deletenodes proto

insertFromLast proto , a:dword

insertatpos proto , a: dword , b:dword

deletefrompos proto , b:dword

\_size = 100 ; 25 nodes of dword

NULL = 0

count dword 0

head dword 0

nextptr dword ?

currnode dword ?

handler dword ?

tempoo dword ?

dwFlags DWORD HEAP\_ZERO\_MEMORY

str1 byte "Node can not be created!" ,0

str2 byte "Node successfully created! ",0

str3 byte "All nodes successfully destroyed! ",0

str4 byte "No node available.",0

str5 byte "First node successfully destroyed.",0

str6 byte "Last node successfully destroyed.",0

str7 byte "Last node successfully created.",0

str8 byte "Node inserted at position: ",0

str9 byte "No node available at position: ",0

str10 byte "Node deleted from position: ",0

.code

main PROC

invoke getprocessheap

mov handler , eax

mov ebx,0

invoke insertnode , 4

invoke insertnode , 3

invoke insertnode , 2

;invoke insertnode , 4

call print

invoke insertatpos , 5,3

call print

invoke deletefrompos , 3

;call removelast

call print

invoke deletefrompos , 3

;call crlf

;call removefront

call print

;invoke deletenodes

;call print

exit

main endp

;;;;;;;;;;;;;;;;;;;;;Insert Node Function;;;;;;;;;;;;;;;;

insertnode proc , a: dword

cmp head , NULL ; if head is null

je l1

mov ecx, count

mov esi,head

mov ebx,[esi+4]

mov eax,[esi]

mov [esi+4] , eax

l:

add esi,4

mov eax,ebx

mov ebx,[esi+4]

mov [esi+4],eax

loop l

mov esi,head

mov eax,a

mov [esi],eax

jmp quit

l1:

invoke heapalloc , handler , dwFlags , \_size

mov head,eax

mov esi , head

mov nextptr , esi

mov eax,a

mov [esi] ,eax

quit:

mov edx,OFFSET str2

call writestring

call crlf

inc count

ret

insertnode endp

;;;;;;;;;;;;;;;;;;;;;Remove Front Function;;;;;;;;;;;;;;;;

removefront proc

cmp head,NULL

je l1

cmp count,1

je l2

mov esi,head

add esi,4

mov head,esi

dec count

jmp l3

l1:

mov edx,offset str4

call writestring

call crlf

jmp l3

l2:

call deletenodes

l3:

mov edx,offset str5

call writestring

call crlf

ret

removefront endp

;;;;;;;;;;;;;;;;;;;;;Remove Last Function;;;;;;;;;;;;;;;;

removelast proc

local temp:dword

local last:dword

cmp head,NULL

je l1

cmp count,1

je l2

dec count

mov esi,head

mov temp,esi

mov last,esi

l5:

cmp temp,NULL

je l4

mov esi,temp

mov last,esi

add temp,4

jmp l5

l4:

mov last,NULL

jmp l3

l1:

mov edx,offset str4

call writestring

call crlf

jmp l3

l2:

call deletenodes

l3:

mov edx,offset str6

call writestring

call crlf

ret 8

removelast endp

;;;;;;;;;;;;;;;;;;;;;Print Function;;;;;;;;;;;;;;;;

print proc

mov ecx,count

mov esi, head

cmp count,0

je l3

l1:

mov eax,[esi]

call writedec

call crlf

add esi,4

loop l1

jmp l4

l3:

mov edx,offset str4

call writestring

call crlf

l4:

ret

print endp

;;;;;;;;;;;;;;;;;;;;;Delete Nodes Function;;;;;;;;;;;;;;;;

deletenodes proc

INVOKE HeapFree, handler, dwFlags, head

mov edx,OFFSET str3

call writestring

call crlf

mov count,0

mov head,NULL

ret

deletenodes endp

;;;;;;;;;;;;;;;;;;;;;Insert From Last Function;;;;;;;;;;;;;;;;

InsertFromLast proc , a: dword

mov edx,offset str4

call writestring

cmp head , NULL ; if head is null

je lx

mov ecx, count

mov esi , head

ly:

loop ly

mov eax, esi

mov eax, a

mov [esi],eax

jmp quitt

lx:

invoke heapalloc , handler , dwFlags , \_size

mov head,eax

mov esi , head

mov nextptr , esi

mov eax,a

mov [esi] ,eax

quitt:

call crlf

mov edx,OFFSET str7

call writestring

call crlf

inc count

ret

InsertFromLast endp

insertatpos proc , a: dword , b:dword

mov eax,4

mov ebx,b

dec ebx

mul ebx

mov esi,head

add esi,eax

mov tempoo,esi

mov eax,[esi]

cmp eax , 0 ; if that index is null

je l1

mov ecx, count

mov ebx,[esi+4]

mov eax,[esi]

mov [esi+4] , eax

l:

add esi,4

mov eax,ebx

mov ebx,[esi+4]

mov [esi+4],eax

loop l

mov eax,a

mov esi,tempoo

mov [esi],eax

jmp quit

l1:

mov nextptr , esi

mov eax,a

mov [esi] ,eax

quit:

mov edx,OFFSET str8

call writestring

mov eax,b

call writedec

call crlf

inc count

ret

insertatpos endp

deletefrompos proc , b:dword

cmp head,NULL

je nonode

mov eax,4

mov ebx,b

dec ebx

mul ebx

mov esi,head

add esi,eax

mov eax,[esi]

cmp eax , 0 ; if that index is null

je error

mov ecx,count

l1:

mov eax,[esi+4]

mov [esi],eax

add esi,4

loop l1

mov edx,OFFSET str10

call writestring

mov eax,b

call writedec

call crlf

dec count

jmp quit

nonode:

mov edx,OFFSET str4

call writestring

call crlf

jmp quit

error:

mov edx,OFFSET str9

call writestring

mov eax,b

call writedec

call crlf

jmp quit

quit:

ret

deletefrompos endp

end main

Binary Search Tree

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

include irvine32.inc

.data

insertnodeBST proto , a:dword

delete proto ,x:dword,y:dword,z:dword

Searching proto , q:dword , e:dword , s:dword

PreOrderTraversal proto ,e:dword

visit proto,l:dword

ltr1 byte "Node successfully created! ",0

ltr2 byte "No node available.",0

ltr3 byte "Tree Deleted!!",0

ltr4 byte "In Order Triversal OF Binary Tree Is:.",0

ltr5 byte "Number Found:.",0

ltr6 byte "Not Found:.",0

ltr7 byte "Search Number :.",0

ltr9 byte "Delete call :.",0

ltr8 byte "::::::::::::::::::BINARY SEARCH TREE :::::::::::::::::::.",0

vsearch dword 0

\_size = 100

NULL = 0

count dword 0

root dword 0

n dword 0

i dword 0

lchild dword 0

rchild dword 0

currnode dword ?

handler dword ?

tempoo dword ?

value dword ?

dwFlags DWORD HEAP\_ZERO\_MEMORY

.code

main proc

invoke getprocessheap

mov handler , eax

mov ebx,0

mov edx,offset ltr8

call writestring

call crlf

invoke insertnodeBST , 34

invoke insertnodeBST , 22

invoke insertnodeBST , 42

invoke insertnodeBST , 10

invoke insertnodeBST , 23

invoke insertnodeBST , 4

push root

push count

call print

call crlf

mov value,10

invoke delete , root , count , value

call print

mov vsearch ,10

invoke Searching , root , count , vsearch

call print

;invoke PreOrderTraversal ,root

call InOrderTraversal

call print

call deletetree

call print

main endp

insertnodeBST proc , a: dword

cmp root ,0

je j1

mov ecx,count

dec ecx

;mov esi,eax

sub esi,4

insert:

mov ebx,[esi]

cmp a,ebx

jg insertleft

jmp insertright

loop insert

jmp quitbst

insertleft:

mov edx,0

cmp [esi+4] , edx

je m2

add esi,4

loop insert

m2:

mov ebx,a

mov lchild,ebx

mov [esi+4],ebx

jmp quitbst

add esi,4

insertRight:

add esi,4

mov edx,0

cmp [esi+8],edx

;sub esi,8

je k2

;add esi,4

loop insert

k2:

mov ebx,a

mov rchild,ebx

mov [esi+4],ebx

mov rchild,ebx

jmp quitbst

j1:

invoke heapalloc , handler , dwFlags , \_size

mov root , eax

mov currnode ,eax

;mov [edx],eax

mov esi,currnode

mov eax,a

mov [esi],eax

quitbst:

mov edx,OFFSET ltr1

call writestring

call crlf

;mov ecx,[esi]

;mov [edx],ecx

add esi,4

;add edx,4

inc count

ret

insertnodeBST endp

print proc

mov ecx , count

mov esi, root

cmp count,0

je j5

j4:

mov eax ,[esi]

call writedec

call crlf

add esi,4

loop j4

jmp j6

j5:

mov edx,offset ltr2

call writestring

call crlf

j6:

ret

print endp

InOrderTraversal proc

push ebx

mov ebx,esp

mov ecx,[ebx+8]

mov edx,offset ltr4

call writestring

call crlf

dec ecx

l1:

push ecx

mov eax,[ebx+12]

;mov lchilld,[ebx+12]

mov esi,eax

l2:

mov eax,[esi]

mov rchild,eax

cmp eax,[esi+4]

jb l3

xchg eax,[esi+4]

mov [esi],eax

l3:

add esi,4

loop l2

pop ecx

loop l1

pop ebp

ret 8

InOrderTraversal endp

DeleteTree Proc

INVOKE HeapFree, handler, dwFlags, root

mov edx,OFFSET ltr3

call writestring

call crlf

mov count,0

mov root,NULL

ret

DeleteTree endp

Delete proc ,t:dword,k:dword ,v:dword

mov edx,offset ltr9

call writestring

call crlf

mov esi,root

mov ecx,k

del:

mov ebx,[esi]

mov eax,v

cmp eax,ebx

je p2

add esi,4

loop del

jmp d

p2:

mov edx,[esi+4]

mov ebx,0

cmp edx,ebx

je p3

cmp [esi+8],ebx

jne bold

mov [esi],edx

jmp d

p3:

mov edx,[esi+8]

mov ebx,0

cmp edx,ebx

je d

bold:

mov ecx,[esi+12]

mov ebx,0

cmp ecx,ebx

jne m

mov ebx,[esi+8]

mov [esi],ebx

d:

dec count

ret

m:

mov ebx,[esi+8]

mov [esi],ebx

ret

delete endp

PreOrderTraversal Proc , w:dword

mov esi,w

mov eax,0

call writedec

cmp ebx,eax

je h

jmp r

h:

ret

r:

mov esi,w

mov ebx,[esi+4]

mov w,ebx

invoke PreOrderTraversal , w

mov ebx,[esi+8]

mov w,ebx

invoke PreOrderTraversal, w

mov ebx,[esi]

mov w,ebx

invoke visit,w

PreOrderTraversal endp

Searching proc , q:dword , e:dword , s:dword

mov esi,q

mov ecx,e

mov ebx,s

mov edx,offset ltr7

call writestring

mov eax,s

call writedec

call crlf

op:

mov eax,[esi]

cmp eax,ebx

je found

add esi,4

loop op

mov edx,offset ltr6

call writestring

call crlf

ret

found:

mov edx,offset ltr5

call writestring

call crlf

ret

searching endp

visit proc , f:dword

mov eax,f

call writedec

call crlf

ret

visit endp

end main