ADVANCE MICROPROCESSOR PROJECT

DHARAMSINH DESAI UNIVERSITY

FACULTY OF TECHNOLOGY

Department of Information Technology
Faculty of Technology, Dharmsinh Desai University
College Road, Nadiad, Gujarat India-387001

Year 2015-2016

PROJECT NAME:

DATA STRUCTURES IN ASSEMBLY

PREPARED BY:

` KRUTI JAYESH RAVAL

GUIDED BY:

PROF. R.S.CHHAJED

INDEX

Project on Data Structures

DESCRIPTION:

The Data structures presented in this code also includes the output in Video Graphics Adapter (VGA) mode.
The output is presented pictorially in VGA-320X200 graphics mode (256 colours).All the different function calls used in the following codes are mentioned below:

To shift to VGA (320X200) graphics mode, interrupt used is:

```
mov ah,00h
mov al,13h
int 10h.
```

To set cursor position:

INT 10h / **AH = 2** - set cursor position.
input:
DH = row.
DL = column.
BH = page number (0..7).

to get cursor position and size:

INT 10h / **AH = 03h** - get cursor position and size.
input:
BH = page number.
return:
DH = row.
DL = column.
CH = cursor start line.
CL = cursor bottom line.

To set pixel color:

INT 10h / **AH = 0Ch** - change color for a single pixel.
input:
AL = pixel color

CX = column.
DX = row.

To write string:

INT 10h / **AH = 13h** - write string

input:
AL = write mode:
  bit 0: update cursor after writing;
  bit 1: string contains attributes.
BH = page number.
BL = attributes if string contains only characters (bit 1 of AL is zero).
CX = number of characters in string (attributes are not counted).
DL,DH = column, row at which to start writing.
ES:BP points to string to be printed.

To set video mode:

INT 10h / **AH = 0** - set video mode.
input:
AL = desired video mode.
these video modes are supported:
00h - text mode. 40x25. 16 colors. 8 pages.
03h - text mode. 80x25. 16 colors. 8 pages.
13h - graphical mode. 40x25. 256 colors. 320x200 pixels. 1 page.

To read a character from standard input:

INT 21h / **AH=1** - read character from standard input, with echo, result is stored in **AL**.if there is no character in the keyboard buffer, the function waits until any key is pressed**.**

To write character to standard output:

INT 21h / **AH=2** - write character to standard output.
entry: **DL** = character to write, after execution **AL = DL**.

To get character input:

INT 21h / **AH=7** - character input without echo to AL.
if there is no character in the keyboard buffer, the function waits until any key is pressed.

To get input of a string:

INT 21h / **AH=0Ah** - input of a string to **DS:DX**, fist byte is buffer size, second byte is number of chars actually read. this function does **not** add '$' in the end of string. to print using **INT 21h** / **AH=9** you must set dollar character at the end of it and start printing from address **DS:DX + 2**.

To get output of a string:

INT 21h / **AH=9** - output of a string at **DS:DX**. String must be terminated by '**$**'.

To create file:

INT 21h / **AH= 3Ch** - create or truncate file.
CX = file attributes:

```
mov cx, 0 ; normal - no attributes.
mov cx, 1 ; read-only.
mov cx, 2 ; hidden.
mov cx, 4 ; system
mov cx, 7 ; hidden, system and read-only!
mov cx, 16 ; archive
```

DS:DX -> ASCIZ filename. returns:
CF clear if successful, **AX** = file handle.
CF set on error **AX** = error code.

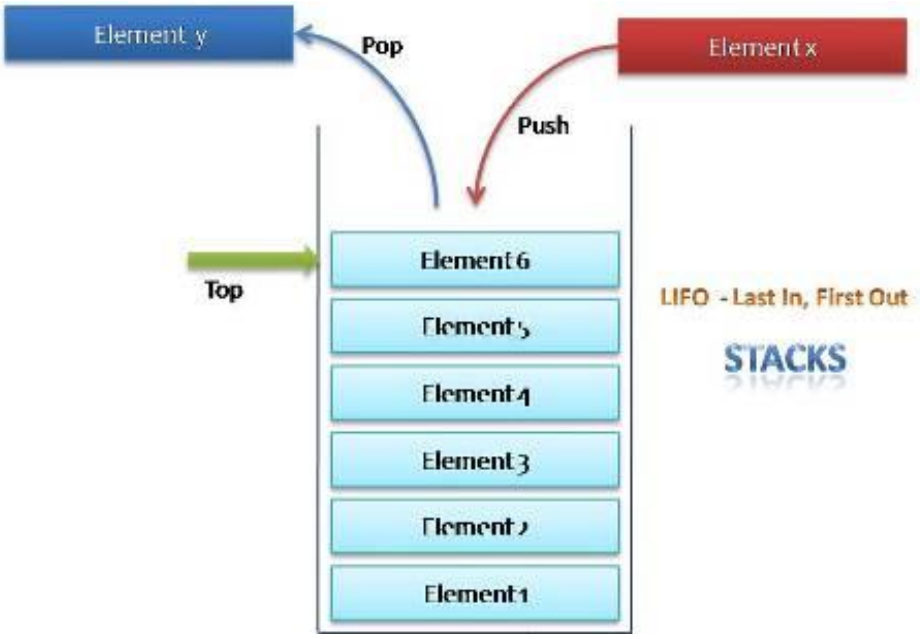
To write to a file:

INT 21h / **AH= 40h** - write to file.
BX = file handle.
CX = number of bytes to write.
DS:DX -> data to write. return:
CF clear if successful; **AX** = number of bytes actually written.
CF set on error; **AX** = error code.

STACK:

A Stack is data structure in which addition of new element or deletion of existing element always takes place at a same end. This end is known as the top of the stack. That means that it is possible to remove elements from a stack in reverse order from the insertion of elements into the stack.

One other way of describing the stack is as a **last in, first out (LIFO)** abstract data type and linear data structure.



CODE FOR STACK:

Kruti.asm(stack)

```
1   0000                        data segment
2   0000  0A 0D 43 48 4F 4F53+  str1 db 0Ah, 0Dh,"CHOOSE OPERATION", 0Ah, 0Dh, '$'
3         45 20 4F 50 45 52     41+
4         54 49 4F 4E 0A 0D     24
5   0015  0A 0D 50 20 46 4F           52+  str2 db 0Ah, 0Dh,"P FOR PUSH",0ah,0dh,'$'
6         20 50 55 53 48 0A     0D+
7         24
8   0024  0A 0D 52 20 46 4F           52+  str3 db 0Ah, 0Dh,"R FOR POP",0ah,0dh,'$'
9         20 50 4F 50 0A 0D     24
10  0032  0A 0D 44 20 46 4F           52+  str8 db 0Ah, 0Dh,"D FOR DISPLAY",0ah,0dh,'$'
11        20 44 49 53 50 4C     41+
12        59 0A 0D 24
13  0044  0A 0D 41 6E 79 20           6B+  str9 db 0Ah, 0Dh,"Any key for exit",0ah,0dh,'$'
14        65 79 20 66 6F 72     20+
15        65 78 69 74 0A 0D     24
16  0059  0A 0D 45 6E 74 65           72+  str4 db 0Ah, 0Dh,"Enter value to be pushed",0ah,0dh,'$'
17        20 76 61 6C 75 65     20+
18        74 6F 20 62 65 20     70+
19        75 73 68 65 64 0A     0D+
20        24
21  0076  0A 0D 4F 76 65 72           66+  str5 db 0Ah, 0Dh,"Overflow",0ah,0dh,'$'
```

```
22        6C 6F 77 0A 0D 24
23   0083  0A 0D 44 6F 20 79          6F+  str6 db 0Ah, 0Dh,"Do you want to continue?
     (y/n)",0ah,0dh,'$'
24        75 20 77 61 6E 74     20+
25        74 6F 20 63 6F 6E     74+
26        69 6E 75 65 3F 20     28+
27        79 2F 6E 29 0A 0D     24
28   00A6  0A 0D 45 6D 70 74          79+  str7 db 0Ah, 0Dh,"Empty stack",0ah,0dh,'$'
29        20 73 74 61 63 6B     0A+
30        0D 24
31   00B6  0A 0D 50 6F 70 70          69+  str10 db 0Ah, 0Dh,"Popping..$"
32        6E 67 2E 2E 24
33   00C2  0A 0D 0A 0D 20 45          6E+  msg2 db 10, 13, 10, 13, ' Enter name of file to be      created
     : $'
34        74 65 72 20 6E 61     6D+
35        65 20 6F 66 20 66     69+
36        6C 65 20 74 6F 20     62+
37        65 20 63 72 65 61     74+
38        65 64 20 3A 20 24
39   00EB  0A 0D 0A 0D 20 46          69+  msg16 db 10, 13, 10, 13, '    File successfully created :)$'
40        6C 65 20 73 75 63     63+
41        65 73 73 66 75 6C     6C+
42        79 20 63 72 65 61     74+
43        65 64 20 3A 29 24
44
45
46   010D  01*(00)                choose db 1 dup(0)
47   010E  01*(00)                top db 1 dup(0)
48   010F  01*(00)                temp db 1 dup(0)
49   0110  01*(00)                t db 1 dup(0)
50   0111  64*(00)                array db 100 dup(0)
51   0175  00                     position db 00h
52   0176  00                     no_of_elements db 00h
53   0177  01*(00)                maxsize db    1 dup(0)
54   0178  50 00 50*(00)          buffer db 80, 0, 80 dup(0)
55   01CA  50 00 50*(00)          buffer1 db     80, 0, 80 dup(0)
56   021C  50 00 50*(00)          buffer2 db     80, 0, 80 dup(0)
57   026E                         data ends
58   0000                         my_stack segment stack
59   0000  64*(0000)                              dw    100 dup(0)
60   00C8                              top_stack label word
61   00C8                         my_stack ends
62   0000                         code segment
63                                     assume cs:code, ds:data
64   0000  B8 0000s                    start: mov     ax,data
65   0003  8E D8                              mov   ds,ax
66   0005  C6 06 010Er 00                     mov   top,00
67   000A  C6 06 010Fr 00                     mov   temp,00
```

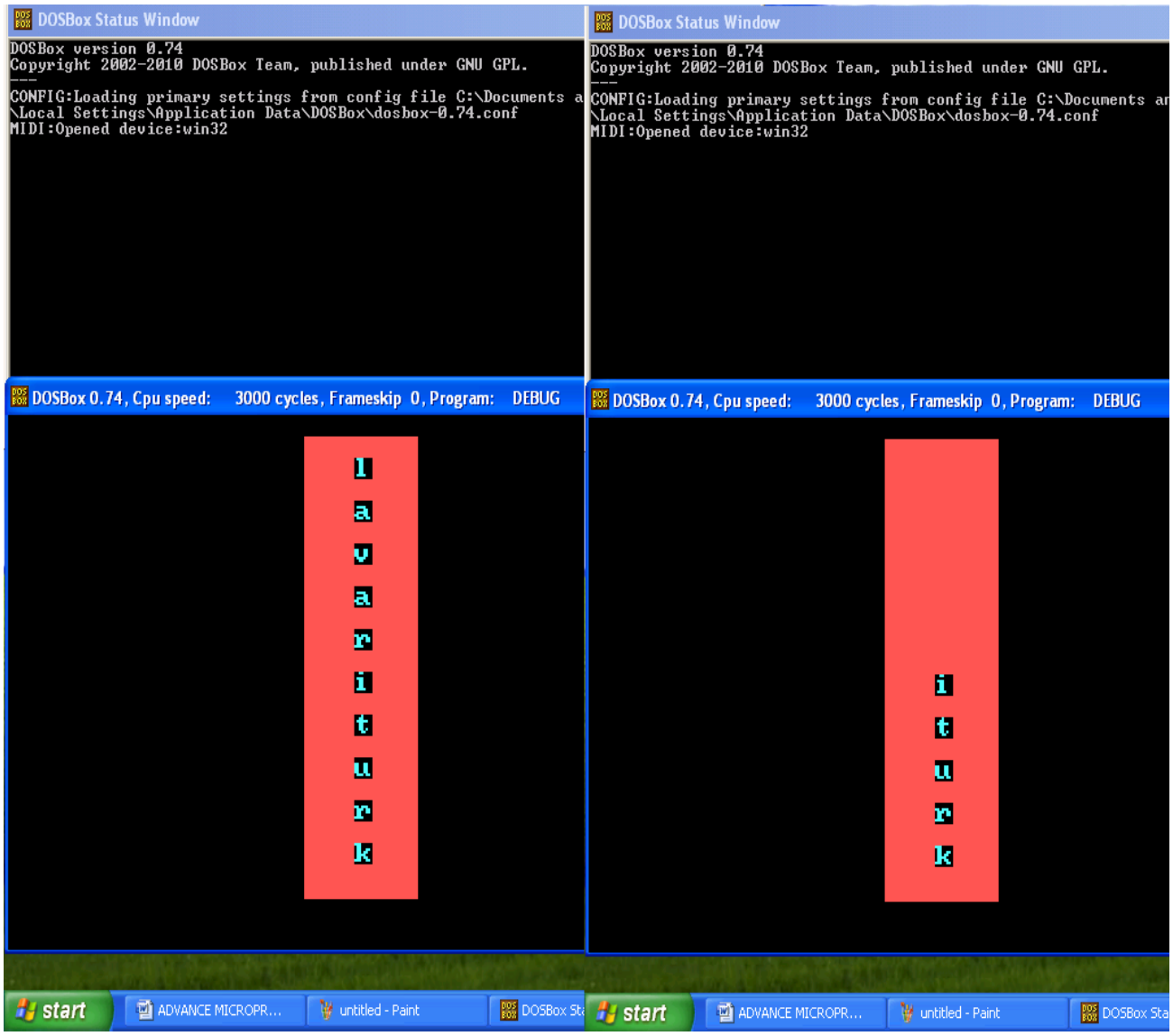
```
 68   000F  C6 06 0177r 0A                mov  maxsize,10
 69   0014  BE 0111r                     lea  si, array
 70   0017  BF 0111r                     lea  di, array
 71   001A                          menu:
 72   001A  BA 0000r                     mov  dx, offset str1
 73   001D  B4 09                        mov  ah,09h
 74   001F  CD 21                        int  21h
 75   0021  BA 0015r                     mov  dx, offset str2
 76   0024  B4 09                        mov  ah,09h
 77   0026  CD 21                        int  21h
 78   0028  BA 0024r                     mov  dx, offset str3
 79   002B  B4 09                        mov  ah,09h
 80   002D  CD 21                        int  21h
 81   002F  BA 0032r                     mov  dx, offset str8
 82   0032  B4 09                        mov  ah,09h
 83   0034  CD 21                        int  21h
 84   0036  BA 0044r                     mov  dx, offset str9
 85   0039  B4 09                        mov  ah,09h
 86   003B  CD 21                        int  21h
 87   003D  B4 01                        mov  ah,01
 88   003F  CD 21                        int  21h
 89   0041  3C 70                        cmp  al, 'p'
 90   0043  74 0B                        je p
 91   0045  3C 72                        cmp  al, 'r'
 92   0047  74 3A                        je r
 93   0049  3C 64                        cmp  al, 'd'
 94   004B  74 6A                        je d
 95
 96   004D  E9 00CC                      jmp  e
 97   0050  80 3E 010Er 0A          p:   cmp  top,10
 98   0055  75 09                        jne  x
 99   0057  BA 0076r                     mov  dx, offset str5
100   005A  B4 09                        mov  ah,09h
101   005C  CD 21                        int  21h
102   005E  EB BA                        jmp  menu
103   0060  BA 0059r                x:   mov  dx, offset str4
104   0063  B4 09                        mov  ah,09h
105   0065  CD 21                        int  21h
106   0067  B4 01                        mov  ah,01
107   0069  CD 21                        int  21h
108   006B  88 04                        mov  [si],al
109   006D  46                           inc  si
110   006E  FE 06 010Er                  inc  top
111   0072  BA 0083r                     mov  dx, offset str6
112   0075  B4 09                        mov  ah,09h
113   0077  CD 21                        int  21h
114   0079  B4 01                        mov  ah,01
115   007B  CD 21                        int  21h
```

```
116  007D  3C 79                          cmp  al, 'y'
117  007F  74 CF                          je p
118  0081  EB 97                          jmp  menu
119  0083  80 3E 010Er 00          r:     cmp  top,00
120  0088  75 09                          jne  y
121  008A  BA 00A6r                       mov  dx, offset str7
122  008D  B4 09                          mov  ah,09h
123  008F  CD 21                          int  21h
124  0091  EB 87                          jmp  menu
125  0093  BA 00B6r                y:     mov  dx, offset str10
126  0096  B4 09                          mov  ah,09h
127  0098  CD 21                          int  21h
128  009A  4E                             dec  si
129  009B  FE 0E 010Er                    dec  top
130  009F  8A 14                          mov  dl,[si]
131  00A1  B4 02                          mov  ah,02
132  00A3  CD 21                          int  21h
133  00A5  BA 0083r                       mov  dx, offset str6
134  00A8  B4 09                          mov  ah,09h
135  00AA  CD 21                          int  21h
136  00AC  B4 01                          mov  ah,01
137  00AE  CD 21                          int  21h
138  00B0  3C 79                          cmp  al, 'y'
139  00B2  74 CF                          je r
140  00B4  E9 FF63                        jmp  menu
141  00B7  B4 00                    d:    mov  ah,00h
142  00B9  B0 13              mov al,13h
143  00BB  CD 10              int 10h
144
145  00BD  B9 0082            mov cx,130
146  00C0  BA 0008            mov dx,8
147  00C3  B0 0C              mov al,0ch
148  00C5  B4 0C              mov ah,0ch
149  00C7                     again_draw:
150  00C7  CD 10              int 10h
151  00C9  41                 inc cx
152  00CA  81 F9 00B4         cmp cx,180
153  00CE  75 F7              jne again_draw
154  00D0  B9 0082            mov cx,130
155  00D3  42                 inc dx
156  00D4  81 FA 00B5         cmp dx,181
157  00D8  75 ED              jne again_draw
158
159
160
161  00DA  B8 0000s           mov ax,data
162  00DD  8E C0              mov es,ax
163  00DF  A0 010Er           mov al,top
```

```
00E2  A2 0176r            mov no_of_elements,al
00E5  BD 0111r            mov bp,offset array
00E8  C6 06 0175r 14      mov position,20
00ED                      again_push:
00ED  B0 01                       mov al,01h
00EF  B7 00                       mov bh,0h
00F1  B3 0B                       mov bl,0bh
00F3  B9 0001                     mov cx,01h
00F6  B2 13                       mov dl,19
00F8  8A 36 0175r                 mov dh,position
00FC  B4 13                       mov ah,13h
00FE  CD 10                       int 10h
0100  45                          inc bp
0101  FE 0E 0175r                 dec position
0105  FE 0E 0175r                 dec position
0109  FE 0E 0176r                 dec no_of_elements
010D  75 DE                       jnz again_push
010F  B4 07                       mov ah,07
0111  CD 21                       int 21h
0113  B4 00                       mov ah,00h
0115  B0 00                       mov al,00h
0117  CD 10                       int 10h
0119  E9 FEFE                     jmp menu


011C                      e:
011C  BA 00C2r                lea dx, msg2        ; module for creating a file
011F  B4 09                     mov ah, 09h     ; a string      on screen
0121  CD 21                      int 21h

0123  C6 06 01CAr 50             mov [buffer1], 80     ; first string
0128  BA 01CAr                   lea dx, buffer1
012B  B4 0A                      mov ah, 0ah              ; read string from keyboard
012D  CD 21                      int 21h
012F  8A 1E 01CBr                mov bl, buffer1[1]
0133  B7 00                      mov bh, 0
0135  83 C3 02                   add bx, 2
0138  C6 87 01CAr 00             mov buffer1[bx],    0     ; read name of file to be

013D  BA 01CCr                   lea dx, buffer1[2]
0140  B9 0000                    mov cx, 0
0143  B4 3C                      mov ah, 3ch       ; create      the file
0145  CD 21                      int 21h

0147  8B D8               mov bx,ax        ;move file     handle
0149  BE 010Er                   lea si,top
014C  8B 04                       mov ax,[si]
014E  8B C8                       mov cx,ax
```

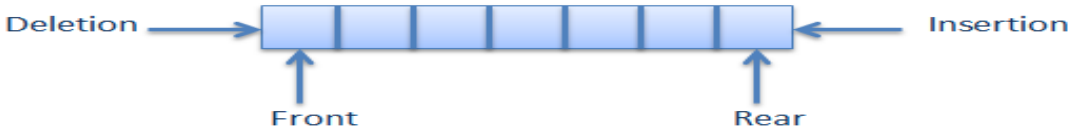
```
212   0150  BA 0111r                lea dx, array[0]
213   0153  B4 40                   mov ah, 40h       ; write to the file
214   0155  CD 21                   int 21h
215   0157  BA 00EBr                lea dx,msg16
216   015A  B4 09                   mov ah, 09h      ; a string       on screen
217   015C  CD 21                    int 21h
218   015E  EB 01 90                jmp ee
219
220   0161  CC                  ee:    int 3
221
222
223   0162                      code ends
224                             end start
```

Output:


DOSBox Status Window
DOSBox version 0.74
Copyright 2002-2010 DOSBox Team, published under GNU GPL.
CONFIG:Loading primary settings from config file C:\Documents a
\Local Settings\Application Data\DOSBox\dosbox-0.74.conf
MIDI:Opened device:win32
DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: DEBUG
l a v a r i t u r k
start
ADVANCE MICROPR...
untitled - Paint
DOSBox Sta
DOSBox Status Window
DOSBox version 0.74
Copyright 2002-2010 DOSBox Team, published under GNU GPL.
CONFIG:Loading primary settings from config file C:\Documents an
\Local Settings\Application Data\DOSBox\dosbox-0.74.conf
MIDI:Opened device:win32
DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: DEBUG
i t u r k
start
ADVANCE MICROPR...
untitled - Paint
DOSBox Sta

QUEUE:

Queue is a kind of abstract data type where items are inserted one end (rear end) known as enqueueoperation and deteted from the other end(front end) known as dequeue operation.
This makes the queue a **First-In-First-Out (FIFO)** data structure.
The queue performs the function of a buffer.



CODE FOR QUEUE:

q.asm

```
 1    0000                          data segment
 2    0000  0A 0D 43 48 4F 4F53+  str1 db 0ah,0dh,"CHOOSE OPERATION",0ah,0dh,'$'
 3          45 20 4F 50 45 52     41+
 4          54 49 4F 4E 0A 0D     24
 5    0015  0A 0D 50 20 46 4F           52+  str2 db 0ah,0dh,"P FOR PUSH",0ah,0dh,'$'
 6          20 50 55 53 48 0A     0D+
 7          24
 8    0024  0A 0D 52 20 46 4F           52+  str3 db 0ah,0dh,"R FOR POP",0ah,0dh,'$'
 9          20 50 4F 50 0A 0D     24
10    0032  0A 0D 44 20 46 4F           52+  str8 db 0ah,0dh,"D FOR DISPLAY",0ah,0dh,'$'
11          20 44 49 53 50 4C     41+
12          59 0A 0D 24
13    0044  0A 0D 41 6E 79 20           6B+  str9 db 0ah,0dh,"Any key for exit",0ah,0dh,'$'
14          65 79 20 66 6F 72     20+
15          65 78 69 74 0A 0D     24
16    0059  0A 0D 45 6E 74 65           72+  str4 db 0ah,0dh,"Enter value to be  pushed",0ah,0dh,'$'
17          20 76 61 6C 75 65     20+
18          74 6F 20 62 65 20     70+
19          75 73 68 65 64 0A     0D+
20          24
21    0076  0A 0D 4F 76 65 72           66+  str5 db 0ah,0dh,"Overflow",0ah,0dh,'$'
22          6C 6F 77 0A 0D 24
23    0083  0A 0D 44 6F 20 79           6F+  str6 db 0ah,0dh,"Do you want to continue?
(y/n)",0ah,0dh,'$'
24          75 20 77 61 6E 74     20+
25          74 6F 20 63 6F 6E     74+
26          69 6E 75 65 3F 20     28+
27          79 2F 6E 29 0A 0D     24
```

```
00A6  0A 0D 45 6D 70 74          79+  str7 db 0ah,0dh,"Empty stack",0ah,0dh,'$'
      20 73 74 61 63 6B     0A+
      0D 24
00B6  0A 0D 50 6F 70 70          69+  str10 db 0Ah, 0Dh,"Popping..$"
      6E 67 2E 2E 24

00C2  0A 0D 0A 0D 20 45          6E+  msg2 db 10, 13, 10, 13, ' Enter name of file to be     created
: $'
      74 65 72 20 6E 61     6D+
      65 20 6F 66 20 66     69+
      6C 65 20 74 6F 20     62+
      65 20 63 72 65 61     74+
      65 64 20 3A 20 24
00EB  0A 0D 0A 0D 20 46          69+  msg16 db 10, 13, 10, 13, '   File successfully created :)$'
      6C 65 20 73 75 63     63+
      65 73 73 66 75 6C     6C+
      79 20 63 72 65 61     74+
      65 64 20 3A 29 24

010D  01*(00)                    choose db 1 dup(0)
010E  01*(00)                    front db 1     dup(0)
010F  01*(00)                    rear db 1 dup(0)
0110  01*(00)                    temp db 1 dup(0)
0111  01*(00)                    f db 1 dup(0)
0112  64*(00)                    array db 100 dup(0)
0176  00                         position db 00h
0177  00                         no_of_elements db 00
0178  00                         counter db    00
0179  01*(00)                    maxsize db    1 dup(0)

017A  50 00 50*(00)              buffer db 80, 0, 80 dup(0)
   58          01CC  50 00 50*(00)          buffer1 db     80, 0, 80 dup(0)
021E  50 00 50*(00)              buffer2 db     80, 0, 80 dup(0)

0270                             data ends
0000                             my_stack segment stack
0000  64*(0000)                                  dw    100 dup(0)
00C8                                    top_stack label word
00C8                             my_stack ends
0000                             code segment
                                        assume cs:code, ds:data
0000  B8 0000s                          start: mov     ax,data
0003  8E D8                                    mov   ds,ax
0005  C6 06 010Er 00                           mov   front,00
000A  C6 06 010Fr 00                           mov   rear,00
000F  BE 0112r                                 lea     si,array
0012  BF 0112r                                 lea     di,array
0015  BA 0000r                          menu:  mov   dx, offset str1
```

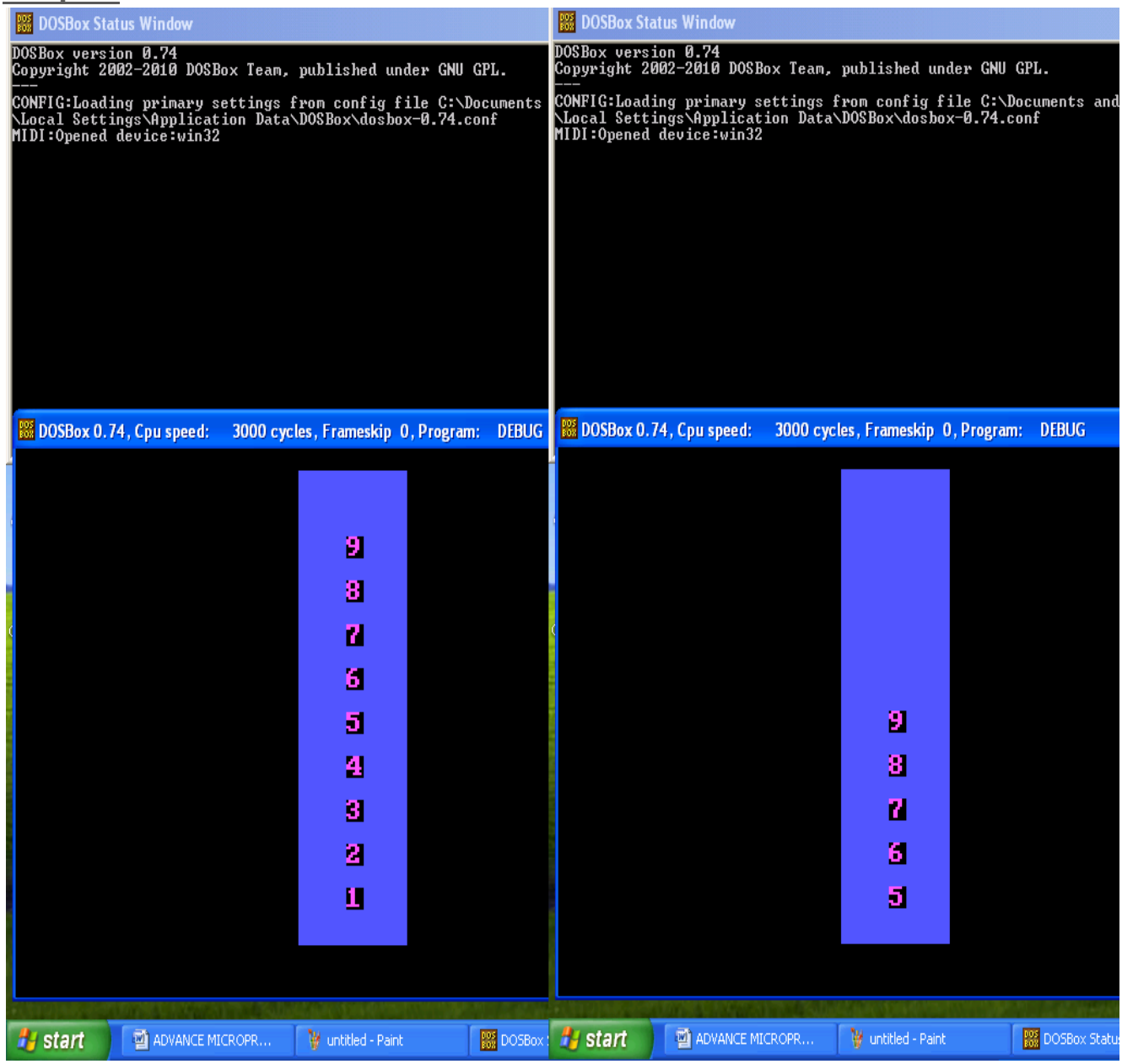
```
 75   0018  B4 09                        mov  ah,09h
 76   001A  CD 21                        int  21h
 77   001C  BA 0015r                     mov  dx, offset str2
 78   001F  B4 09                        mov  ah,09h
 79   0021  CD 21                        int  21h
 80   0023  BA 0024r                     mov  dx, offset str3
 81   0026  B4 09                        mov  ah,09h
 82   0028  CD 21                        int  21h
 83   002A  BA 0032r                     mov  dx, offset str8
 84   002D  B4 09                        mov  ah,09h
 85   002F  CD 21                        int  21h
 86   0031  BA 0044r                     mov  dx, offset str9
 87   0034  B4 09                        mov  ah,09h
 88   0036  CD 21                        int  21h
 89   0038  B4 01                        mov  ah,01
 90   003A  CD 21                        int  21h
 91   003C  3C 70                        cmp  al, 'p'
 92   003E  74 0B                        je p
 93   0040  3C 72                        cmp  al, 'r'
 94   0042  74 3A                        je r
 95   0044  3C 64                        cmp  al, 'd'
 96   0046  74 6E                        je d
 97   0048  E9 00D3                      jmp  e
 98   004B  80 3E 010Fr 0A       p:      cmp  rear,10
 99   0050  75 09                        jne  x
100   0052  BA 0076r                   mov  dx, offset str5
101   0055  B4 09                        mov  ah,09h
102   0057  CD 21                        int  21h
103   0059  EB BA                        jmp  menu
104   005B  BA 0059r             x:      mov  dx, offset str4
105   005E  B4 09                        mov  ah,09h
106   0060  CD 21                        int  21h
107   0062  B4 01                        mov  ah,01
108   0064  CD 21                        int  21h
109   0066  FE 06 010Fr                  inc  rear
110   006A  88 04                        mov  [si],al
111   006C  46                           inc  si
112   006D  BA 0083r                     mov  dx, offset str6
113   0070  B4 09                        mov  ah,09h
114   0072  CD 21                        int  21h
  115         0074  B4 01                    mov  ah,01
116   0076  CD 21                        int  21h
117   0078  3C 79                        cmp  al, 'y'
118   007A  74 CF                        je p
119   007C  EB 97                        jmp  menu
120   007E  A0 010Er             r:      mov  al, front
121   0081  8A 16 010Fr                  mov  dl, rear
122   0085  3A C2                        cmp  al,dl
```

```
0087  75 09                          jne   y
0089  BA 00A6r                       mov   dx, offset str7
008C  B4 09                          mov   ah,09h
008E  CD 21                          int   21h
0090  EB 83                          jmp   menu
0092  BA 00B6r                 y:    mov   dx, offset str10
0095  B4 09                          mov   ah,09h
0097  CD 21                          int   21h
0099  FE 06 010Er                     inc   front
009D  8A 15                          mov   dl,[di]
009F  B4 02                          mov   ah, 02
00A1  CD 21                          int   21h
00A3  47                             inc   di
00A4  BA 0083r                       mov   dx, offset str6
00A7  B4 09                          mov   ah,09h
00A9  CD 21                          int   21h
00AB  B4 01                          mov   ah,01
00AD  CD 21                          int   21h
00AF  3C 79                          cmp   al, 'y'
00B1  74 CB                          je r
00B3  E9 FF5F                        jmp   menu
00B6                            d:
00B6  B4 00                          mov   ah,00h
00B8  B0 13                mov al,13h
00BA  CD 10                int 10h

00BC  B9 0082              mov cx,130
00BF  BA 0008              mov dx,8
00C2  B0 09                mov al,09h
00C4  B4 0C                mov ah,0ch
00C6                       again_draw:
00C6  CD 10                int 10h
00C8  41                   inc cx
00C9  81 F9 00B4           cmp cx,180
00CD  75 F7                jne again_draw
00CF  B9 0082              mov cx,130
00D2  42                   inc dx
00D3  81 FA 00B5           cmp dx,181
00D7  75 ED                jne again_draw


00D9  B8 0000s             mov ax,data
00DC  8E C0                mov es,ax
00DE  A0 010Fr             mov al,rear
00E1  2A 06 010Er          sub al,front
00E5  A2 0177r             mov no_of_elements,al
00E8  8B EF                mov bp,di
```

```
00EA  C6 06 0176r 14        mov position,20
00EF                        again_push:
00EF  B0 01                     mov al,01h
00F1  B7 00                     mov bh,0h
00F3  B3 0B                     mov bl,0dh
00F5  B9 0001                   mov cx,01h
00F8  B2 13                     mov dl,19
00FA  8A 36 0176r               mov dh,position
00FE  B4 13                     mov ah,13h
0100  CD 10                     int 10h
0102  45                        inc bp
0103  FE 0E 0176r               dec position
0107  FE 0E 0176r               dec position
010B  FE 0E 0177r               dec no_of_elements
010F  75 DE                     jnz again_push
0111  B4 07                     mov ah,07
0113  CD 21                     int 21h
0115  B4 00                     mov ah,00h
0117  B0 00                     mov al,00h
0119  CD 10                     int 10h
011B  E9 FEF7                   jmp menu

011E                            e:
011E  BA 00C2r                  lea dx, msg2        ; module for creating a file
0121  B4 09                     mov ah, 09h     ; a string      on screen
0123  CD 21                      int 21h

0125  C6 06 01CCr 50             mov [buffer1], 80      ; first string
012A  BA 01CCr                   lea dx, buffer1
012D  B4 0A                      mov ah, 0ah                ; read string from keyboard
012F  CD 21                      int 21h
0131  8A 1E 01CDr                mov bl, buffer1[1]
0135  B7 00                      mov bh, 0
0137  83 C3 02                   add bx, 2
013A  C6 87 01CCr 00             mov buffer1[bx],     0     ; read name of file to be

013F  BA 01CEr                   lea dx, buffer1[2]
0142  B9 0000                    mov cx, 0
0145  B4 3C                      mov ah, 3ch       ; create      the file
0147  CD 21                      int 21h

0149  8B D8                 mov bx,ax        ;move file     handle
014B  A0 010Fr                  mov al,rear
014E  2A 06 010Er               sub al,front
0152  B4 00                     mov ah,00
0154  8B C8                     mov cx,ax
0156  8D 15                     lea dx, [di]
0158  B4 40                     mov ah, 40h       ; write to the file
```

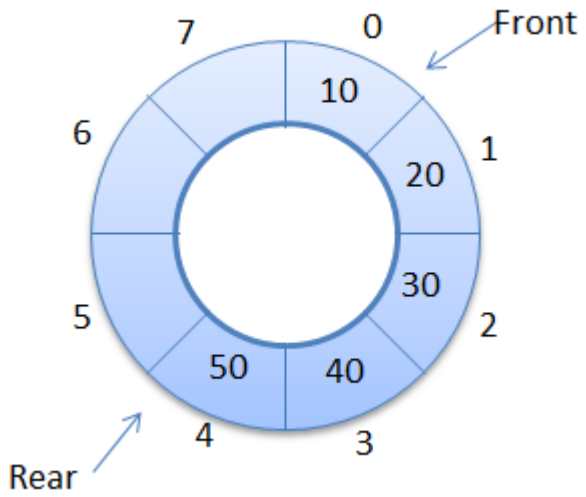
```
219   015A  CD 21                   int 21h
220   015C  BA 00EBr                lea dx,msg16
221   015F  B4 09                   mov ah, 09h     ; a string      on screen
222   0161  CD 21                    int 21h
223   0163  EB 01 90                jmp ee
224
225   0166  CC                    ee:    int 3
226
227   0167                        code ends
228                               end star
```

Output:



CIRCULAR QUEUE:

A circular queue is an abstract data type that contains a collection of data which allows addition of data at the end of the queue and removal of data at the beginning of the queue. Circular queues have a fixed size.
Circular queue follows **FIFO** principle. Queue items are added at the rear end and the items are deleted at front end of the circular queue.



CODE FOR QUEUE:

cirq.asm

```
 1    0000                          data segment
 2
 3    0000  64*(00)                 array db 100 dup(0)
 4
 5    0064  0A 0D 43 48 4F 4F53+  str1 db 0ah,0dh,"CHOOSE OPERATION",0ah,0dh,'$'
 6          45 20 4F 50 45 52     41+
 7          54 49 4F 4E 0A 0D     24
 8    0079  0A 0D 50 20 46 4F           52+  str2 db 0ah,0dh,"P FOR PUSH",0ah,0dh,'$'
 9          20 50 55 53 48 0A     0D+
10          24
11    0088  0A 0D 51 20 46 4F           52+  str3 db 0ah,0dh,"Q FOR POP",0ah,0dh,'$'
12          20 50 4F 50 0A 0D     24
13    0096  0A 0D 44 20 46 4F           52+  str8 db 0ah,0dh,"D FOR DISPLAY",0ah,0dh,'$'
14          20 44 49 53 50 4C     41+
15          59 0A 0D 24
16    00A8  0A 0D 41 6E 79 20           6B+  str9 db 0ah,0dh,"Any key for exit",0ah,0dh,'$'
```

```
17        65 79 20 66 6F 72     20+
18        65 78 69 74 0A 0D     24
19    00BD  0A 0D 45 6E 74 65           72+  str4 db 0ah,0dh,"Enter value to be  pushed",0ah,0dh,'$'
20        20 76 61 6C 75 65     20+
21        74 6F 20 62 65 20     70+
22        75 73 68 65 64 0A     0D+
23        24
24    00DA  0A 0D 4F 76 65 72           66+  str5 db 0ah,0dh,"Overflow",0ah,0dh,'$'
25        6C 6F 77 0A 0D 24
26    00E7  0A 0D 44 6F 20 796F+  str6 db 0ah,0dh,"Do you want to continue? (y/n)",0ah,0dh,'$'
27        75 20 77 61 6E 74     20+
28        74 6F 20 63 6F 6E     74+
29        69 6E 75 65 3F 20     28+
30        79 2F 6E 29 0A 0D     24
31    010A  0A 0D 45 6D 70 74           79+  str7 db 0ah,0dh,"Empty queue",0ah,0dh,'$'
32        20 71 75 65 75 65     0A+
33        0D 24
34    011A  0A 0D 50 6F 70 70           69+  str10 db 0ah,0dh,"Popping...",'$'
35        6E 67 2E 2E 2E 24
36    0127  01*(00)                choose db 1 dup(0)
37    0128  01*(00)                front db 1      dup(0)
38    0129  01*(00)                rear db 1 dup(0)
39    012A  01*(00)                newr db 1 dup(0)
40    012B  01*(00)                maxsize db    1 dup (0)
41    012C  01*(00)                f db 1 dup(0)
42    012D  01*(00)                re      db 1 dup(0)
43    012E  01*(00)                t db 1 dup(0)
44
45    012F  05 08 05 02 05         qx     db 5,8,5,2,5
46    0134  04 06 08 06 04         qy     db 4,6,8,6,4
47    0139  00                     qpointer db 00
48    013A  04                     maxsize_S db 04h
49
50    013B                         data ends
51    0000                         code segment
52                                          assume cs:code, ds:data,es:data
53    0000  B8 0000s                         start:mov     ax,data
54    0003  8E D8                                   mov   ds,ax
55    0005  8E C0                                                 mov es,ax
56    0007  C6 06 012Br 05                        mov   maxsize,05
57    000C  C6 06 0128r 00                        mov   front,00h
58    0011  C6 06 0129r 00                        mov   rear,00h
59    0016  BA 0064r                       menu:  mov   dx, offset str1
60    0019  B4 09                                 mov   ah,09h
61    001B  CD 21                                 int     21h
62    001D  BA 0079r                              mov   dx, offset str2
63    0020  B4 09                                 mov   ah,09h
64    0022  CD 21                                 int     21h
```

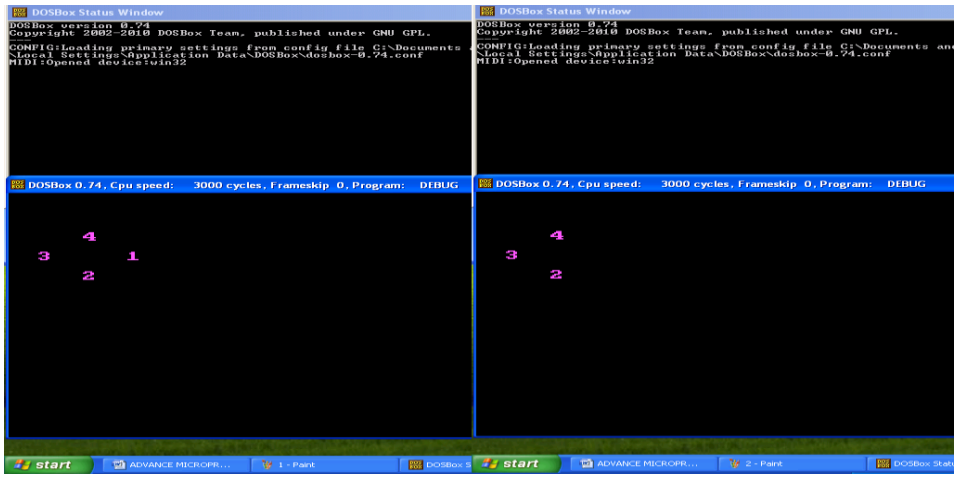
```
   65    0024  BA 0088r                    mov  dx, offset str3
   66    0027  B4 09                       mov  ah,09h
   67    0029  CD 21                       int  21h
   68    002B  BA 0096r                    mov  dx, offset str8
   69    002E  B4 09                       mov  ah,09h
   70    0030  CD 21                       int  21h
   71    0032  BA 00A8r                    mov  dx, offset str9
   72    0035  B4 09                       mov  ah,09h
   73    0037  CD 21                       int  21h
   74    0039  B4 01                       mov  ah,01
   75    003B  CD 21                       int  21h
   76    003D  3C 70                       cmp  al, 'p'
   77    003F  74 0E                       je p
   78    0041  3C 71                       cmp  al, 'q'
   79    0043  74 60                       je q
   80    0045  3C 64                       cmp  al, 'd'
   81    0047  74 03                       je call_d
   82    0049  E9 00A9                     jmp  e
   83    004C  E9 0147             call_d: jmp  db2
   84    004F  A0 0129r            p:      mov  al,rear
   85    0052  FE C0                       inc  al
   86    0054  B4 00                       mov  ah,00
   87    0056  F6 36 012Br                 div  maxsize          ;calculate
         newr=(rear+1)%maxsize
   88    005A  88 26 012Ar                 mov  newr,ah
   89    005E  A0 012Ar                    mov  al, newr
   90    0061  8A 16 0128r                 mov  dl,front
   91    0065  8A 26 0129r                 mov  ah,rear
   92    0069  3A C2                       cmp  al,dl            ;if front=newrear
then overflow
   93    006B  75 09                       jne  x
   94    006D  BA 00DAr                    mov  dx,offset str5
   95    0070  B4 09                       mov  ah,09h
   96    0072  CD 21                       int  21h
   97    0074  EB A0                       jmp  menu
   98    0076                      x:
   99    0076  8A 26 012Ar                 mov  ah,newr                ;else
rear=newrear
  100    007A  88 26 0129r                 mov  rear,ah
  101    007E  BA 00BDr                    mov  dx,offset str4         ;ask user for
value
  102    0081  B4 09                       mov  ah,09h
  103    0083  CD 21                       int  21h
  104    0085  B4 01                       mov  ah,01
  105    0087  CD 21                       int  21h
  106    0089  8A 1E 0129r                 mov  bl,rear
  107    008D  B7 00                       mov  bh,00
```

```
   108   008F  88 87 0000r                       mov  [array+bx],al
;q[rear]=value
   109                                 ;                    cmp front,0ffh
   110                                 ;                    jne skip_inc
   111                                 ;                    inc front
   112                                 ;                skip_inc:
   113   0093  BA 00E7r                           mov dx,offset str6                ;ask
to continue
   114   0096  B4 09                              mov  ah,09h
   115   0098  CD 21                              int    21h
   116   009A  B4 01                              mov  ah,01
   117   009C  CD 21                              int    21h
   118   009E  3C 79                              cmp  al, 'y'
   119   00A0  74 AD                              je p
   120   00A2  E9 FF71                            jmp   menu
   121   00A5  A0 0128r                    q:     mov  al, front
   122   00A8  8A 16 0129r                        mov  dl, rear
   123   00AC  3A C2                              cmp  al,dl                        ;if front=rear, then
queue   empty
   124   00AE  75 0A                              jne    y
   125   00B0  BA 010Ar                           mov  dx, offset str7
   126   00B3  B4 09                              mov  ah,09h
   127   00B5  CD 21                              int    21h
   128   00B7  E9 FF5C                            jmp   menu
   129   00BA  BA 011Ar                    y:     mov  dx, offset str10
   130   00BD  B4 09                              mov  ah,09h
   131   00BF  CD 21                              int    21h
   132   00C1  A0 0128r                           mov  al,front
   133   00C4  FE C0                              inc    al
   134   00C6  B4 00                              mov  ah,0
   135   00C8  F6 36 012Br                        div    maxsize                    ;else
front=(front+1)%maxsize
   136   00CC  88 26 0128r                        mov  front,ah
   137   00D0  8A 1E 0128r                        mov  bl,front
   138   00D4  B7 00                              mov  bh,00
   139   00D6  8A 97 0000r                        mov  dl,[array+bx]
   140   00DA  C6 87 0000r 00                     mov  [array+bx],00                ;make that
         element           0
   141   00DF  B4 02                              mov  ah, 02
   142   00E1  CD 21                              int    21h
   143   00E3  BA 00E7r                           mov  dx, offset str6              ;ask to continue
   144   00E6  B4 09                              mov  ah,09h
   145   00E8  CD 21                              int    21h
   146   00EA  B4 01                              mov  ah,01
   147   00EC  CD 21                              int    21h
   148   00EE  3C 79                              cmp  al,'y'
   149   00F0  74 B3                              je q
   150   00F2  E9 FF21                            jmp   menu
```

```
   151   00F5  CC                    e:      int   3
   152   00F6  A0 0128r               d:      mov   al,front
   153   00F9  A2 012Cr                       mov   f,al
   154   00FC  A0 0129r                       mov   al,rear
   155   00FF  A2 012Dr                       mov   re,al
   156   0102  A0 012Cr                       mov   al,f
   157   0105  8A 16 012Dr                    mov   dl,re
   158   0109  3A C2                          cmp   al,dl                ;if front=rear, queue
empty
   159   010B  75 0A                          jne   zz
   160   010D  BA 010Ar                       mov   dx,offset str7
   161   0110  B4 09                          mov   ah,09h
   162   0112  CD 21                          int   21h
   163   0114  E9 FEFF                        jmp   menu
   164   0117                         zz:
   165   0117  A0 012Cr                       mov   al,f
   166   011A  8A 16 012Dr                    mov   dl,re
   167   011E  3A C2                          cmp   al,dl
   168   0120  73 26                          jnc   za
   169   0122  FE 06 012Cr            z:      inc   f                    ;f<r
   170   0126  8A 1E 012Cr                    mov   bl,f
   171   012A  B7 00                          mov   bh,00                ;print simply from
i=front to i=rear
   172   012C  8A 97 0000r                    mov   dl,[array+bx]
   173   0130  B4 02                          mov   ah,02
   174   0132  CD 21                          int   21h
   175   0134  B2 20                          mov   dl,32
   176   0136  B4 02                          mov   ah,02
   177   0138  CD 21                          int   21h
   178   013A  A0 012Cr                       mov   al,f
   179   013D  8A 16 012Dr                    mov   dl,re
   180   0141  3A C2                          cmp   al,dl
   181   0143  75 DD                          jne   z
   182   0145  E9 FECE                        jmp   menu
   183   0148  FE 06 012Cr            za:     inc   f                    ;f>r
   184   014C  8A 1E 012Cr                    mov   bl,f                 ;print from i=front
to i=maxsize
   185   0150  B7 00                          mov   bh,00
   186   0152  8A 97 0000r                    mov   dl,[array+bx]
   187   0156  B4 02                          mov   ah,02
   188   0158  CD 21                          int   21h
   189   015A  B2 20                          mov   dl,32
   190   015C  B4 02                          mov   ah,02
   191   015E  CD 21                          int   21h
   192   0160  A0 012Cr                       mov   al,f
   193   0163  8A 16 012Br                    mov   dl,maxsize
   194   0167  3A C2                          cmp   al,dl
   195   0169  75 DD                          jne   za
```

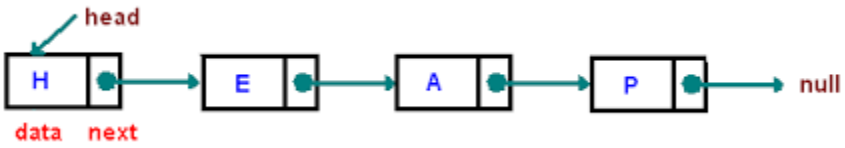
```
   196   016B  C6 06 012Er 00                    mov  t,00
   197   0170  8A 1E 012Er                zb:    mov  bl,t                    ;and then from i=0
         to i=rear
   198   0174  B7 00                             mov  bh,00
   199   0176  8A 97 0000r                       mov  dl,[array+bx]
   200   017A  B4 02                             mov  ah,02
   201   017C  CD 21                             int  21h
   202   017E  B2 20                             mov  dl,32
   203   0180  B4 02                             mov  ah,02
   204   0182  CD 21                             int  21h
   205   0184  A0 012Er                          mov  al,t
   206   0187  8A 16 012Dr                       mov  dl,re
   207   018B  FE 06 012Er                       inc  t
   208   018F  3A C2                             cmp  al,dl
   209   0191  75 DD                             jne  zb
   210   0193  E9 FE80                           jmp  menu
   211
   212
   213   0196  A0 0128r                          db2:  mov  al,front
   214   0199  A2 012Cr                          mov  f,al
   215   019C  A0 0129r                          mov  al,rear
   216   019F  A2 012Dr                          mov  re,al
   217   01A2  A0 012Cr                          mov  al,f
   218   01A5  8A 16 012Dr                       mov  dl,re
   219   01A9  3A C2                             cmp  al,dl                   ;if front=rear, queue
empty
   220   01AB  75 03                             jne  cont
   221   01AD  E9 FE66                                  jmp menu
   222                                          ; jmp z
   223   01B0                                    cont:
   224                                                  ;clearing screen
   225   01B0  B8 0600                                  mov ax,0600h
   226   01B3  B7 00                                    mov bh,00h
   227   01B5  B9 0000                                  mov cx,0
   228   01B8  B6 48                                    mov dh,48h
   229   01BA  B2 79                                    mov dl,79h
   230   01BC  CD 10                                    int 10h
   231                                                  ;ends clearing screen
   232   01BE  B8 0013                                  mov ax,0013h
   233   01C1  CD 10                                    int 10h
   234
   235   01C3  8A 0E 0128r                              mov cl,front
   236   01C7  B5 00                                    mov ch,00
   237   01C9  8B F1                                    mov si,cx
   238   01CB  C6 06 0139r 02                           mov qpointer,02h
   239   01D0  B0 01                             show_q:mov al,01h
   240   01D2  B7 00                                    mov bh,0h
   241   01D4  B3 96                                    mov bl, 0dh
```

```
242   01D6  B9 0001                              mov cx,01h
243   01D9  8A 94 012Fr                         mov dl,qx[si]
244   01DD  8A B4 0134r                         mov dh,qy[si]
245   01E1  BD 0000r                            mov bp,offset array
246   01E4  03 EE                               add bp,si
247   01E6  B4 13                               mov ah,13h
248   01E8  CD 10                               int 10h
249   01EA  46                                  inc si
250   01EB  8B C6                               mov ax,si
251   01ED  F6 36 012Br                         div maxsize
252   01F1  8A C4                               mov al,ah
253   01F3  B4 00                               mov ah,00
254   01F5  8B F0                               mov si,ax
255   01F7  80 3E 0139r 01                      cmp qpointer,01h
256   01FC  74 0C                               je      skip_poi
257   01FE  3A 06 0129r                         cmp al,rear
258   0202  75 CC                               jne show_q
259   0204  FE 0E 0139r                         dec qpointer
260   0208  75 C6                               jnz show_q
261
262   020A                              skip_poi:
263   020A  B4 07                               mov ah,07h
264   020C  CD 21                               int 21h
265
266   020E  B8 0003                             mov ax,0003h
267   0211  CD 10                               int 10h
268
269   0213  E9 FE00                             jmp menu
270
271   0216  CC                                  int 3
272
273
274
275   0217                      code ends
276                             end start
```



LINKED LIST:

Singly linked lists contain nodes which have a data field as well as a 'next' field, which points to the next node in line of nodes. Operations that can be performed on singly linked lists include insertion, deletion and traversal.



A singly linked list whose nodes contain two fields:
Any data value and a link to the next node

Code for linklist:

linkl.asm

```
    1    0000                        data segment
    2    0000  0A 0D 43 48 4F 4F       53+  str1 db 0ah,0dh,"CHOOSE OPERATION",0ah,0dh,'$'
    3          45 20 4F 50 45 52  41+
    4          54 49 4F 4E 0A 0D  24
    5    0015  0A 0D 45 4E 54 45       52+  str2 db 0ah,0dh,"ENTER NUMBER OF NODE TO BE
DELETED",0AH,0DH,'$'
    6          20 4E 55 4D 42 45  52+
    7          20 4F 46 20 4E 4F  44+
    8          45 20 54 4F 20 42  45+
    9          20 44 45 4C 45 54  45+
   10          44 0A 0D 24
   11    003C  0A 0D 43 55 52 52       45+  str11 db 0ah,0dh,"CURRENT NO OF NODES ARE:
         ",'$'
   12          4E 54 20 4E 4F 20  4F+
   13          46 20 4E 4F 44 45  53+
   14          20 41 52 45 3A 20  24
   15    0058  0A 0D 52 20 46 4F       52+  str3 db 0ah,0dh,"R        FOR POP",0ah,0dh,'$'
   16          20 50 4F 50 0A 0D  24
   17    0066  0A 0D 44 20 46 4F       52+  str8 db 0ah,0dh,"D        FOR DISPLAY",0ah,0dh,'$'
   18          20 44 49 53 50 4C  41+
   19          59 0A 0D 24
   20    0078  0A 0D 41 6E 79 20       6B+  str9 db 0ah,0dh,"Any key for exit",0ah,0dh,'$'
   21          65 79 20 66 6F 72  20+
   22          65 78 69 74 0A 0D  24
   23    008D  0A 0D 45 6E 74 65       72+  str4 db 0ah,0dh,"Enter value to be
         pushed",0ah,0dh,'$'
```

```
   24        20 76 61 6C 75 65  20+
   25        74 6F 20 62 65 20  70+
   26        75 73 68 65 64 0A  0D+
   27        24
   28   00AA  0A 0D 4F 76 65 72        66+  str5 db 0ah,0dh,"Overflow",0ah,0dh,'$'
   29        6C 6F 77 0A 0D 24
   30   00B7  0A 0D 44 6F 20 79        6F+  str6 db 0ah,0dh,"Do you want to continue?
(y/n)",0ah,0dh,'$'
   31        75 20 77 61 6E 74  20+
   32        74 6F 20 63 6F 6E  74+
   33        69 6E 75 65 3F 20  28+
   34        79 2F 6E 29 0A 0D  24
   35   00DA  0A 0D 45 6D 70 74        79+  str7 db 0ah,0dh,"Empty list",0ah,0dh,'$'
   36        20 6C 69 73 74 0A  0D+
   37        24
   38   00E9  0A 0D 50 6F 70 70        69+  str10 db 0ah,0dh,"Popping        ",0ah,0dh,'$'
   39        6E 67 20 20 0A 0D  24
   40   00F7  0A 0D 50 20 46 4F        52+  str12 db 0ah,0dh,"P FOR PUSH",0ah,0dh,'$'
   41        20 50 55 53 48 0A  0D+
   42        24
   43   0106  01*(00)              choose db 1 dup(0)
   44   0107  01*(00)              max db 1 dup(0)
   45   0108  01*(00)              num db 1 dup(0)
   46   0109  01*(00)              cnt db 1 dup(0)
   47   010A  01*(00)              ele db 1 dup(0)
   48   010B  01*(0000)                  top dw 1 dup (0)
   49   010D  01*(0000)                  prev dw 1 dup(0)
   50   010F  01*(0000)                  temp dw 1 dup(0)
   51   *111                       node struc
   52   *000  01*(00)              val db 0
   53   *001  01*(0000)                  next dw 0
   54   *003                       node ends
   55   0111  04*(00 0000)         n node 4 dup(<>)
   56
   57   011D  1E*(00)              linklist db 30 dup(0)
   58   013B  00                   pointerl db 0
   59
   60
   61   013C                       data ends
   62   0000                       code segment
   63                                   assume cs:code,ds:data,es:data
   64   0000  B8 0000s                       start: mov   ax,data
   65   0003  8E D8                          mov ds,ax
   66   0005  8E C0                                  mov es,ax
```

```
 67   0007  C6 06 0107r 04                mov max,04
 68   000C  C6 06 0108r 00                mov num,00
 69   0011  C6 06 010Ar 00                mov ele,00
 70   0016  BA 0000r                       menu:  movdx, offset str1
 71   0019  B4 09                         mov ah,09h
 72   001B  CD 21                         int    21h
 73   001D  BA 00F7r                              mov dx, offset str12
 74   0020  B4 09                         mov ah,09h
 75   0022  CD 21                         int    21h
 76   0024  BA 0058r                              mov dx, offset str3
 77   0027  B4 09                         mov ah,09h
 78   0029  CD 21                         int    21h
 79   002B  BA 0066r                              mov dx, offset str8
 80   002E  B4 09                         mov ah,09h
 81   0030  CD 21                         int    21h
 82   0032  BA 0078r                              mov dx, offset str9
 83   0035  B4 09                         mov ah,09h
 84   0037  CD 21                         int    21h
 85   0039  B4 07                         mov ah,07
 86   003B  CD 21                         int    21h
 87   003D  3C 70                         cmp al, 'p'
 88   003F  74 0B                         je p
 89   0041  3C 72                         cmp al, 'r'
 90   0043  74 4E                         je call_r
 91   0045  3C 64                         cmp al, 'd'
 92   0047  74 17                         je call_d
 93   0049  E9 015E                       jmp  e
 94   004C  A0 0108r                      p:        mov al, num
 95   004F  8A 16 0107r                   mov dl, max
 96   0053  3A C2                         cmp al,dl
 97   0055  75 0C                         jne   x
 98                                       ;overflow
 99   0057  BA 00AAr                              mov dx,offset str5
100   005A  B4 09                         mov ah,09
101   005C  CD 21                         int    21h
102   005E  EB B6                         jmp  menu
103   0060  E9 0148                 call_d: jmp   d
104   0063                          x:        ;ask for value to be entered
105   0063  BA 008Dr                              mov dx, offset str4
106   0066  B4 09                         mov ah,09h
107   0068  CD 21                         int    21h
108   006A  B4 01                         mov ah,01
109   006C  CD 21                         int    21h
110   006E  80 3E 0108r 00                cmp num,00
```

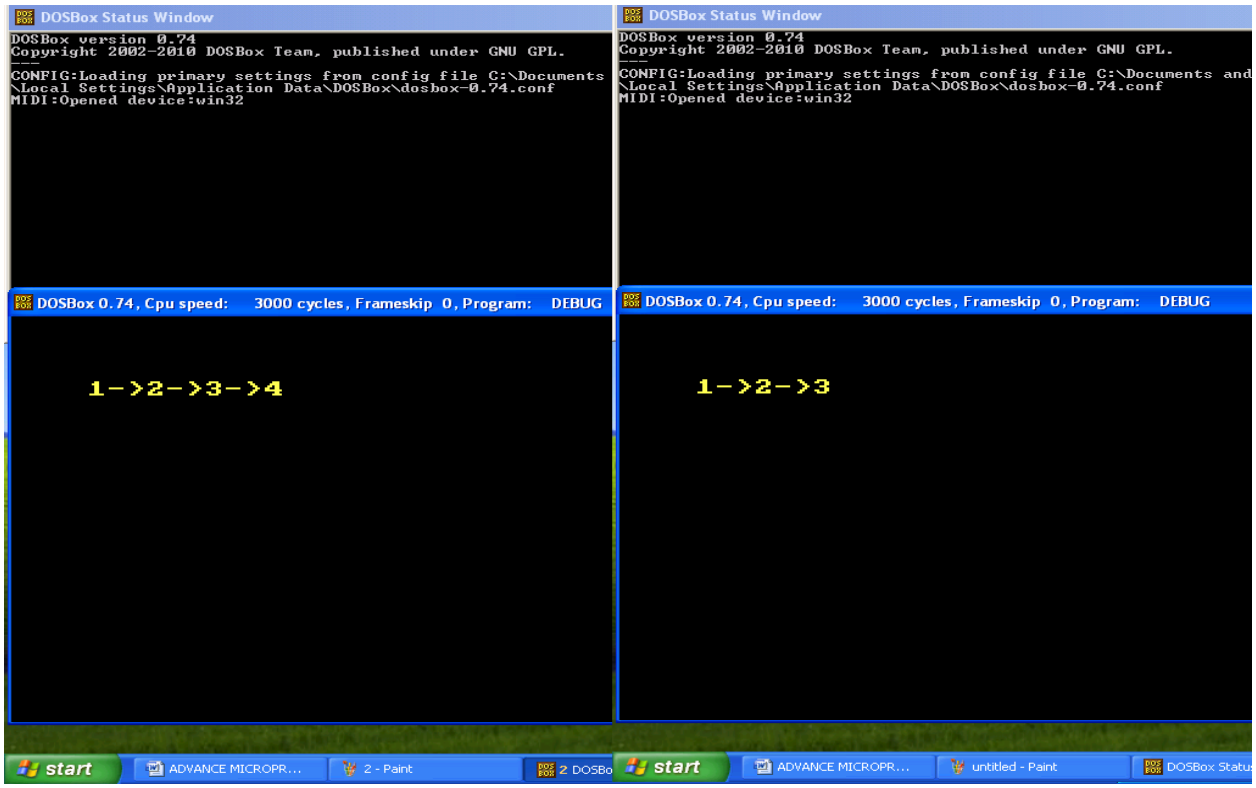
```
111  0073  75 21                          jne   x1
112                                       ;this if list is empty
113  0075  A2 0111r                             mov n[0].val,al
114  0078  C7 06 010Br 0111r                   mov top, offset n[0]                ;set
top. point to    first node
115  007E  FE 06 0108r                        inc   num
116  0082  BA 00B7r                           x2:      mov dx, offset str6          ;ask
to continue
117  0085  B4 09                              mov ah,09h
118  0087  CD 21                              int   21h
119  0089  B4 01                              mov ah,01
120  008B  CD 21                              int   21h
121  008D  3C 79                              cmp al, 'y'
122  008F  74 BB                              je p
123  0091  EB 83                              jmp  menu
124  0093  EB 34 90                    call_r: jmp   r
125  0096                              x1:      ;this if list is not empty
126  0096  8B 1E 010Br                        mov bx,top
127  009A  89 1E 010Fr                        mov temp,bx
128  009E  8B 36 010Fr                        mov si,temp
129  00A2  8B 3E 010Fr                        mov di,temp
130  00A6  46                                 inc   si
131  00A7  89 3E 010Fr                 x12:   mov   temp,di
132  00AB  8B 3C                              mov di,[si]                         ;go to the
      next node till              +
133                                cur->next!=NULL
134  00AD  83 FF 00                                 cmp  di,0000
135  00B0  74 05                              je x11
136  00B2  8B F7                              mov si,di
137  00B4  46                                 inc   si
138  00B5  EB F0                              jmp  x12
139  00B7  83 C6 02                           x11:   add   si,2
140  00BA  88 04                              mov [si],al                         ;add new
node val and a ptr from the   +
141                                prev node
142  00BC  8B 3E 010Fr                        mov di, temp
143  00C0  47                                 inc   di
144  00C1  89 35                              mov [di],si
145  00C3  FE 06 0108r                        inc   num
146  00C7  EB B9                              jmp  x2
147
148  00C9  80 3E 0108r 00              r:     cmp num,00
149  00CE  75 0A                              jne   y
150                                       ;empty
```

```
   151  00D0  BA 00DAr                          mov dx, offset str7
   152  00D3  B4 09                       mov ah,09h
   153  00D5  CD 21                       int   21h
   154  00D7  E9 FF3C               m:    jmp  menu
   155  00DA                        y:    ;not empty
   156  00DA  BA 003Cr                          mov dx, offset str11
   157  00DD  B4 09                       mov ah,09h
   158  00DF  CD 21                       int   21h
   159  00E1  8A 16 0108r                 mov dl,num
   160  00E5  80 C2 30                    add  dl,30h
   161  00E8  B4 02                       mov ah,02
   162  00EA  CD 21                       int   21h
   163                                    ;ask for the index od node to be deleted
   164  00EC  BA 0015r                          mov dx, offset str2
   165  00EF  B4 09                       mov ah,09h
   166  00F1  CD 21                       int   21h
   167  00F3  B4 01                       mov ah,01
   168  00F5  CD 21                       int   21h
   169  00F7  A2 010Ar                          mov ele,al
   170  00FA  C6 06 0109r 00              mov cnt,00
   171  00FF  80 3E 010Ar 31              cmp ele,31h
   172  0104  75 16                       jne   y1
   173                                    ; this if first ele       is to be deleted
   174  0106  8B 36 010Br                 mov si, top
   175  010A  8B 3E 010Br                 mov di, top
   176  010E  46                          inc   si
   177  010F  8B 3C                       mov di,[si]
   178  0111  89 3E 010Br                 mov top,di
;simply make top point to next +
   179                          (second) node
   180  0115  FE 0E 0108r                 dec  num
   181  0119  EB 7D 90                          jmp  y3
   182  011C  A0 010Ar                    y1:   mov al,ele
   183  011F  8A 16 0108r                 mov dl,num
   184  0123  80 C2 30                    add  dl,30h
   185  0126  3A C2                       cmp al,dl
   186  0128  75 32                       jne   y2
   187                                    ;this if last ele is to be deleted
   188  012A  8B 1E 010Br                 mov bx, top
   189  012E  89 1E 010Fr                 mov temp,bx
   190  0132  8B 36 010Fr                 mov si, temp
   191  0136  8B 3E 010Fr                 mov di, temp
   192  013A  46                          inc   si
   193  013B  89 3E 010Dr           y11:  mov  prev, di
```

```
  194  013F  8B 3C                       mov di, [si]
  195  0141  8B F7                       mov si, di
  196  0143  46                         inc   si
  197  0144  83 3C 00                   cmp  word ptr [si], 0000
;traverse to next node till   +
  198                          cur->next!=NULL
  199  0147  75 F2                       jne   y11                                   ;and
make prev point to NULL
  200  0149  8B 36 010Dr                 mov si, prev
  201  014D  46                         inc   si
  202  014E  C7 04 0000                 mov word ptr [si],0000
  203  0152  FE 0E 0108r                 dec   num
  204  0156  EB 40 90                   jmp  y3
  205  0159  E9 FF6D               back:  jmp   r
  206  015C                        y2:     ; this is ele is in       the middle
  207  015C  C6 06 0109r 31              mov cnt,31h
  208  0161  8B 1E 010Br                 mov bx, top
  209  0165  89 1E 010Fr                 mov temp,bx
  210  0169  8B 36 010Fr                 mov si, temp
  211  016D  8B 3E 010Fr                 mov di, temp
  212  0171  46                         inc   si
  213  0172  89 3E 010Dr           y21:   mov   prev, di
  214  0176  8B 3C                       mov di, [si]
  215  0178  8B F7                       mov si, di
  216  017A  46                         inc   si
  217  017B  FE 06 0109r                 inc   cnt
  218  017F  A0 0109r                        mov al, cnt
;set up a cntr and       travers till+
  219                          cntr is equal to index ele
  220  0182  8A 16 010Ar                 mov dl, ele
  221  0186  3A C2                       cmp al,dl
  222  0188  75 E8                       jne   y21
  223  018A  8B 36 010Dr                 mov si, prev                          ;make si
point to prev node
  224  018E  46                         inc   si
  225  018F  47                         inc   di
  226  0190  8B 05                       mov ax,[di]
  227  0192  89 04                       mov [si],ax
;make prev     node point to the   +
  228                          new node
  229  0194  FE 0E 0108r                 dec   num
  230  0198                        y3:     ; ask to continue
  231  0198  BA 00B7r                        mov dx, offset str6
  232  019B  B4 09                       mov ah,09h
```

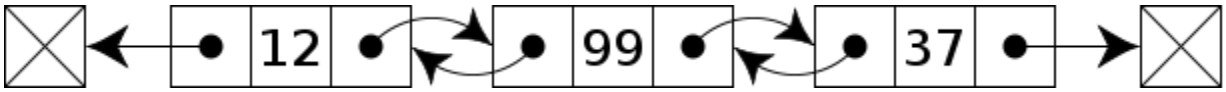
```
233  019D  CD 21                          int    21h
234  019F  B4 01                          mov ah,01
235  01A1  CD 21                          int    21h
236  01A3  3C 79                          cmp al, 'y'
237  01A5  74 B2                          je back
238  01A7  E9 FE6C                        jmp  menu
239  01AA  CC                       e:    int 3
240
241
242  01AB  80 3E 0108r 00                 d:     cmp num,00
243  01B0  74 00                          je cont
244                                                ; jmp z
245  01B2                                 cont:;clearing screen
246  01B2  B8 0600                                mov ax,0600h
247  01B5  B7 00                                  mov bh,00h
248  01B7  B9 0000                                mov cx,0
249  01BA  B6 48                                  mov dh,48h
250  01BC  B2 79                                  mov dl,79h
251  01BE  CD 10                                  int 10h
252                                               ;ends clearing screen
253  01C0  B8 0013                                mov ax,0013h
254  01C3  CD 10                                  int 10h
255
256  01C5  8A 0E 0108r                            mov cl,num
257  01C9  BD 0111r                                     mov bp,offset n
258  01CC  BB 011Dr                                     mov bx,offset linklist
259  01CF  3E: 8A 46          00          copy_ll:mov al,ds:[bp]
260  01D3  88 07                                  mov [bx],al
261  01D5  43                                     inc bx
262  01D6  C6 07 2D                                     mov byte ptr [bx],45
263  01D9  43                                     inc bx
264  01DA  C6 07 3E                                     mov byte ptr [bx],62
265  01DD  43                                     inc bx
266  01DE  83 C5 03                                     add bp,03h
267  01E1  FE C9                                  dec cl
268
269  01E3  75 EA                                  jnz copy_ll
270
271  01E5  8A 0E 0108r                            mov cl,num
272  01E9  FE C9                                  dec cl
273  01EB  B0 03                                  mov al,03h
274  01ED  F6 E1                                  mul cl
275  01EF  FE C0                                  inc al
276  01F1  A2 013Br                                     mov pointerl,al
```

```
277
278  01F4  B0 01                                  mov al,01h
279  01F6  B7 00                                  mov bh,0h
280  01F8  B3 96                                  mov bl, 0Eh
281  01FA  8A 0E 013Br                            mov cl,pointerl
282  01FE  B5 00                                  mov ch,00h
283  0200  B2 04                                  mov dl,4h
284  0202  B6 04                                  mov dh,4h
285  0204  BD 011Dr                                     mov bp,offset linklist

286  0207  B4 13                                  mov ah,13h
287  0209  CD 10                                  int 10h
288
289  020B  B4 07                                  mov ah,07h
290  020D  CD 21                                  int 21h
291
292  020F  B8 0003                                mov ax,0003h
293  0212  CD 10                                  int 10h
294
295  0214  E9 FDFF                                jmp menu
296
297  0217                       code ends
298                             end start
```



DOUBLY LINKED LIST:

In a 'doubly linked list', each node contains, besides the next-node link, a second link field pointing to the 'previous' node in the sequence. The two links may be called 'forward('s') and 'backwards', or 'next' and 'prev'('previous').



A doubly linked list whose nodes contain three fields: an integer value, the link forward to the next node, and the link backward to the previous node

Code for doubly linked list:

Doublyll.asm

```
    1    0000                          data segment
    2    0000  0A 0D 43 48 4F 4F53+  str1 db 0ah,0dh,"CHOOSE OPERATION",0ah,0dh,'$'
    3          45 20 4F 50 45 52     41+
    4          54 49 4F 4E 0A 0D     24
    5    0015  0A 0D 45 4E 54 45           52+  str2 db 0ah,0dh,"ENTER NUMBER OF NODE TO BE
DELETED",0AH,0DH,'$'
    6          20 4E 55 4D 42 45     52+
    7          20 4F 46 20 4E 4F     44+
    8          45 20 54 4F 20 42     45+
    9          20 44 45 4C 45 54     45+
   10          44 0A 0D 24
   11    003C  0A 0D 43 55 52 52           45+  str11 db 0ah,0dh,"CURRENT NO OF NODES ARE:      ",'$'
   12          4E 54 20 4E 4F 20     4F+
   13          46 20 4E 4F 44 45     53+
   14          20 41 52 45 3A 20     24
   15    0058  0A 0D 52 20 46 4F           52+  str3 db 0ah,0dh,"R FOR POP",0ah,0dh,'$'
   16          20 50 4F 50 0A 0D     24
   17    0066  0A 0D 44 20 46 4F           52+  str8 db 0ah,0dh,"D FOR DISPLAY",0ah,0dh,'$'
   18          20 44 49 53 50 4C     41+
   19          59 0A 0D 24
   20    0078  0A 0D 41 6E 79 20           6B+  str9 db 0ah,0dh,"Any key for exit",0ah,0dh,'$'
   21          65 79 20 66 6F 72     20+
   22          65 78 69 74 0A 0D     24
   23    008D  0A 0D 45 6E 74 65           72+  str4 db 0ah,0dh,"Enter value to be  pushed",0ah,0dh,'$'
   24          20 76 61 6C 75 65     20+
   25          74 6F 20 62 65 20     70+
```

```
   26          75 73 68 65 64 0A     0D+
   27          24
   28    00AA  0A 0D 4F 76 65 72           66+  str5 db 0ah,0dh,"Overflow",0ah,0dh,'$'
   29          6C 6F 77 0A 0D 24
   30    00B7  0A 0D 44 6F 20 79           6F+  str6 db 0ah,0dh,"Do you want to continue?
(y/n)",0ah,0dh,'$'
   31          75 20 77 61 6E 74     20+
   32          74 6F 20 63 6F 6E     74+
   33          69 6E 75 65 3F 20     28+
   34          79 2F 6E 29 0A 0D     24
   35    00DA  0A 0D 45 6D 70 74           79+  str7 db 0ah,0dh,"Empty list",0ah,0dh,'$'
   36          20 6C 69 73 74 0A     0D+
   37          24
   38    00E9  0A 0D 50 6F 70 7069+  str10 db 0ah,0dh,"Popping ",0ah,0dh,'$'
   39          6E 67 20 20 0A 0D     24
   40    00F7  0A 0D 50 20 46 4F52+  str12 db 0ah,0dh,"P FOR PUSH",0ah,0dh,'$'
   41          20 50 55 53 48 0A     0D+
   42          24
   43    0106  01*(00)                 choose db 1 dup(0)
   44    0107  01*(00)                 max db 1 dup(0)
   45    0108  01*(00)                 num db 1 dup(0)
   46    0109  01*(00)                 cnt db 1 dup(0)
   47    010A  01*(00)                 ele db 1 dup(0)
   48    010B  01*(0000)                     top dw 1 dup (0)
   49    010D  01*(0000)                     t dw 1 dup(0)
   50    010F  01*(0000)                     temp dw 1 dup(0)
   51    *111                        node struc
   52    *000  01*(0000)                     prev dw 0
   53    *002  01*(00)               val db 0
   54    *003  01*(0000)                     next dw 0
   55    *005                        node ends
   56    0111  04*(0000 00 0000)             n node 4 dup(<>)
   57
   58    0125  1E*(00)               linklist db 30 dup(0)
   59    0143  00                    pointerl db 0
   60
   61    0144                        data ends
   62    0000                        code segment
   63                                      assume cs:code, ds:data,es:data
   64    0000  B8 0000s                    start: mov     ax,data
   65    0003  8E D8                             mov   ds,ax
   66    0005  8E C0                                       mov es,ax
   67    0007  C6 06 0107r 04                    mov   max,04
   68    000C  C6 06 0108r 00                    mov   num,00
   69    0011  C6 06 010Ar 00                    mov   ele,00
   70    0016  BA 0000r                    menu:  mov   dx, offset str1
   71    0019  B4 09                             mov   ah,09h
   72    001B  CD 21                             int     21h
```

```
  73    001D  BA 00F7r                    mov  dx, offset str12
  74    0020  B4 09                      mov  ah,09h
  75    0022  CD 21                      int    21h
  76    0024  BA 0058r                   mov  dx, offset str3
  77    0027  B4 09                      mov  ah,09h
  78    0029  CD 21                      int    21h
  79    002B  BA 0066r                   mov  dx, offset str8
  80    002E  B4 09                      mov  ah,09h
  81    0030  CD 21                      int    21h
  82    0032  BA 0078r                   mov  dx, offset str9
  83    0035  B4 09                      mov  ah,09h
  84    0037  CD 21                      int    21h
  85    0039  B4 07                      mov  ah,07
  86    003B  CD 21                      int    21h
  87    003D  3C 70                      cmp  al, 'p'
  88    003F  74 0B                      je p
  89    0041  3C 72                      cmp  al, 'r'
  90    0043  74 4E                      je call_r
  91    0045  3C 64                      cmp  al, 'd'
  92    0047  74 17                      je call_d
  93    0049  E9 0189                    jmp   e
  94    004C  A0 0108r             p:    mov  al, num
  95    004F  8A 16 0107r                mov  dl, max
  96    0053  3A C2                      cmp  al,dl
  97    0055  75 0C                      jne    x
  98                                     ; overflow
  99    0057  BA 00AAr                   mov  dx,offset str5
 100    005A  B4 09                      mov  ah,09
 101    005C  CD 21                      int    21h
 102    005E  EB B6                      jmp   menu
 103    0060  E9 0173              call_d: jmp    d
 104    0063  BA 008Dr             x:    mov  dx, offset str4
 105    0066  B4 09                      mov  ah,09h
 106    0068  CD 21                      int    21h
 107                                     ;ask for value to be entered
 108    006A  B4 01                      mov  ah,01
 109    006C  CD 21                      int    21h
 110    006E  80 3E 0108r 00             cmp  num,00
 111    0073  75 21                      jne    x1
 112                                     ; this when    list is    empty
 113    0075  A2 0113r                   mov  n[0].val,al                    ;make top
point tofirst node
 114    0078  C7 06 010Br 0111r                  mov  top, offset n[0]
 115    007E  FE 06 0108r                inc    num
 116    0082                       x2:   ; ask to continue
 117    0082  BA 00B7r                   mov  dx, offset str6
 118    0085  B4 09                      mov  ah,09h
 119    0087  CD 21                      int    21h
```

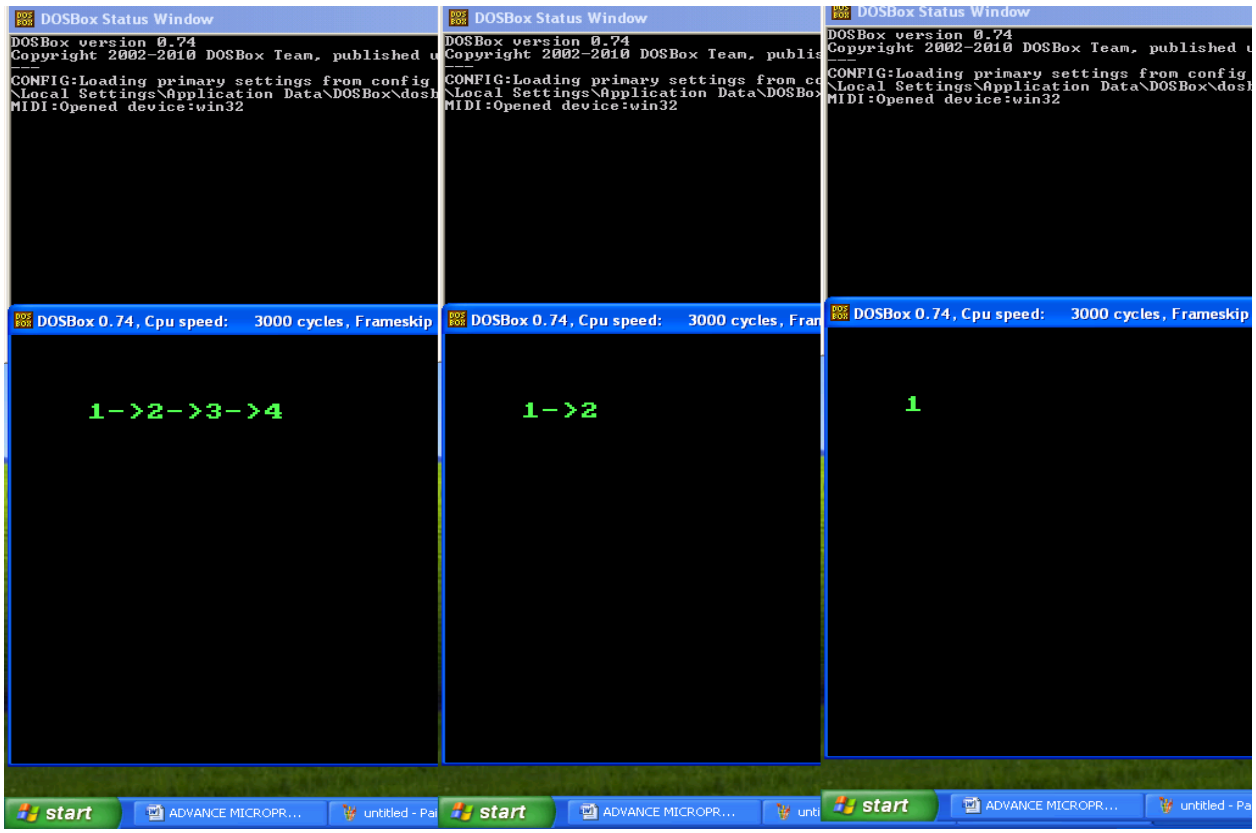
```
   120   0089  B4 01                              mov  ah,01
   121   008B  CD 21                              int    21h
   122   008D  3C 79                              cmp  al, 'y'
   123   008F  74 BB                              je p
   124   0091  EB 83                              jmp   menu
   125   0093  EB 47 90                    call_r: jmp    r
   126   0096                              x1:       ; this when    list is    not empty
   127   0096  8B 1E 010Br                        mov  bx,top
   128   009A  89 1E 010Fr                        mov  temp,bx
   129   009E  8B 36 010Fr                        mov  si,temp
   130   00A2  8B 3E 010Fr                        mov  di,temp
   131   00A6  83 C6 03                           add   si,3
   132   00A9  89 3E 010Fr                 x12:   mov    temp,di
   133   00AD  8B 3C                              mov  di,[si]
   134   00AF  83 FF 00                           cmp  di,0000                              ;travers till
cur->next!=NULL
   135   00B2  74 07                              je x11
   136   00B4  8B F7                              mov  si,di
   137   00B6  83 C6 03                           add   si,3
   138   00B9  EB EE                              jmp   x12
   139   00BB  83 C6 02                    x11:   add     si,2
   140   00BE  89 36 010Dr                        mov  t,si
   141   00C2  83 C6 02                           add   si,2
   142   00C5  88 04                              mov  [si],al                              ;make new
node and          give it    data ip  by  +
   143                                     user
   144   00C7  8B 3E 010Fr                        mov  di, temp
   145   00CB  8B 36 010Dr                        mov  si,t
   146   00CF  89 3C                              mov  [si],di                              ;give the
PREV ptr           to new node
   147   00D1  83 C7 03                           add   di,3
   148   00D4  89 35                              mov  [di],si                              ;give cur
node NEXT ptr to           new node
   149   00D6  FE 06 0108r                        inc    num
   150   00DA  EB A6                              jmp   x2
   151
   152                                     ;          z:         ; if not empty
   153                                     ;                mov bx,top
   154                                     ;                mov temp,bx
   155                                     ;                mov si,temp
   156                                     ;                mov di,temp
   157                                     ;                add si,3
   158                                     ;          z1:    add di,2
   159                                     ;                mov dl,[di]
   160                                     ;                mov ah,02
   161                                     ;                int 21h
   162                                     ;                mov di,[si]
   163                                      ;               mov si,di
```

```
  164                                   ;                 add si,3
  165                                   ;                 cmp di,0000                     ;traverse and print
each value till   +
  166                                   cur->next!=NULL
  167                                   ;                 jne z1
  168                                   ;                 jmp menu
  169   00DC  80 3E 0108r 00             r:      cmp   num,00
  170   00E1  75 0A                              jne   y
  171                                            ; empty
  172   00E3  BA 00DAr                           mov   dx, offset str7
  173   00E6  B4 09                              mov   ah,09h
  174   00E8  CD 21                              int   21h
  175   00EA  E9 FF29                    m:      jmp   menu
  176   00ED                             y:      ; not empty
  177   00ED  BA 003Cr                           mov   dx, offset str11
  178   00F0  B4 09                              mov   ah,09h
  179   00F2  CD 21                              int   21h
  180   00F4  8A 16 0108r                        mov   dl,num
  181   00F8  80 C2 30                           add   dl,30h
  182   00FB  B4 02                              mov   ah,02
  183   00FD  CD 21                              int   21h
  184   00FF  BA 0015r                           mov   dx, offset str2
  185   0102  B4 09                              mov   ah,09h
  186   0104  CD 21                              int   21h
  187   0106  B4 01                              mov   ah,01
  188   0108  CD 21                              int   21h
  189   010A  A2 010Ar                           mov   ele,al
  190   010D  C6 06 0109r 00                     mov   cnt,00
  191   0112  80 3E 010Ar 31                     cmp   ele,31h
  192   0117  75 1C                              jne   y1
  193                                            ; this if first ele          is to be deleted
  194   0119  8B 36 010Br                       mov   si, top
  195   011D  8B 3E 010Br                       mov   di, top
  196   0121  83 C6 03                          add   si,3
  197   0124  8B 3C                             mov   di,[si]
  198   0126  89 3E 010Br                       mov   top,di                                  ;make top
point tonext (second) node
  199   012A  C7 05 0000                        mov   word ptr [di],0000                      ;make PREV
        of the new top 0000
  200   012E  FE 0E 0108r                        dec   num
  201   0132  E9 008E                            jmp   y3
  202   0135  A0 010Ar                  y1:     mov   al,ele
  203   0138  8A 16 0108r                       mov   dl,num
  204   013C  80 C2 30                           add   dl,30h
  205   013F  3A C2                              cmp   al,dl
  206   0141  75 38                              jne   y2
  207                                            ;this if last ele is to be deleted
  208   0143  8B 1E 010Br                       mov   bx, top
```

```
  209   0147  89 1E 010Fr                     mov  temp,bx
  210   014B  8B 36 010Fr                     mov  si, temp
  211   014F  8B 3E 010Fr                     mov  di, temp
  212   0153  83 C6 03                        add  si,3
  213   0156  89 3E 010Dr               y11:  mov  t, di
  214   015A  8B 3C                           mov  di, [si]
  215   015C  8B F7                           mov  si, di
  216   015E  83 C6 03                        add  si,3
  217   0161  83 3C 00                        cmp  word ptr [si], 0000          ;traverse till
cur->next!=NULL
  218   0164  75 F0                           jne  y11
  219   0166  8B 36 010Dr                     mov  si, t
  220   016A  83 C6 03                        add  si,3                         ;(use t to
        keep track of the prev node)
  221   016D  C7 04 0000                      mov  word ptr [si],0000           ;make NEXT
        of second-last node 0000
  222   0171  FE 0E 0108r                     dec  num
  223   0175  EB 4C 90                        jmp  y3
  224   0178  E9 FF61                   back: jmp  r
  225   017B                             y2:   ; this is ele is in      the middle
  226   017B  C6 06 0109r 31                  mov  cnt,31h
  227   0180  8B 1E 010Br                     mov  bx, top
  228   0184  89 1E 010Fr                     mov  temp,bx
  229   0188  8B 36 010Fr                     mov  si, temp
  230   018C  8B 3E 010Fr                     mov  di, temp
  231   0190  83 C6 03                        add  si,3
  232   0193  89 3E 010Dr               y21:  mov  t, di
  233   0197  8B 3C                           mov  di, [si]
  234   0199  8B F7                           mov  si, di
  235   019B  83 C6 03                        add  si, 3
  236   019E  FE 06 0109r                     inc  cnt
  237   01A2  A0 0109r                        mov  al, cnt
  238   01A5  8A 16 010Ar                     mov  dl, ele
  239   01A9  3A C2                           cmp  al,dl                        ;traverse till
count is equal to index
  240   01AB  75 E6                           jne  y21
  241   01AD  8B 36 010Dr                     mov  si, t
  242   01B1  83 C6 03                        add  si,3                         ;si points
        to NEXTof prev  node
  243   01B4  83 C7 03                        add  di,3                         ;di point to
NEXT of node to be deleted
  244   01B7  8B 05                           mov  ax,[di]
  245   01B9  89 04                           mov  [si],ax                      ;make prev
        node point to the next node +
  246                               of   cur node
  247   01BB  8B 3D                           mov  di, [di]                ;di points      to node
        next to  the node to +
  248                               be   deleted
```

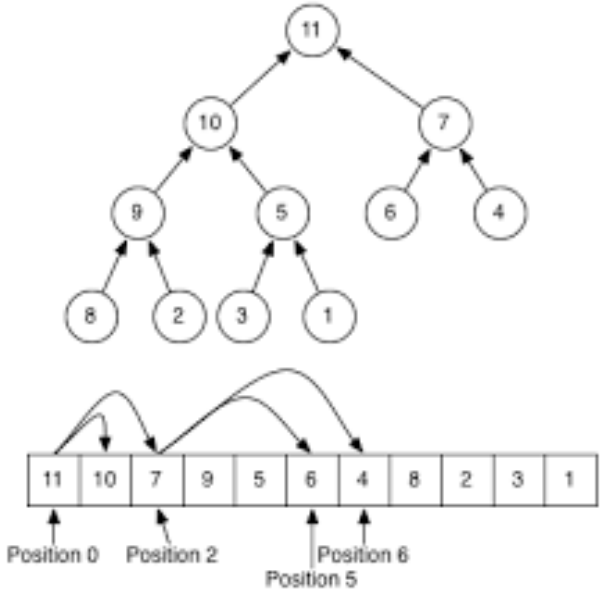
```
  249   01BD  89 35                        mov  [di],si                          ;make it
point to the prev          node
  250   01BF  FE 0E 0108r                  dec    num
  251   01C3                          y3:   ; ask to continue
  252   01C3  BA 00B7r                      mov  dx, offset str6
  253   01C6  B4 09                         mov  ah,09h
  254   01C8  CD 21                         int    21h
  255   01CA  B4 01                         mov  ah,01
  256   01CC  CD 21                         int    21h
  257   01CE  3C 79                         cmp  al, 'y'
  258   01D0  74 A6                         je back
  259   01D2  E9 FE41                       jmp   menu
  260   01D5  CC                      e:    int 3
  261
  262   01D6  80 3E 0108r 00              d:        cmp num,00
  263   01DB  74 00                         je cont
  264                                               ; jmp z
  265   01DD                                  cont:;clearing screen
  266   01DD  B8 0600                                 mov ax,0600h
  267   01E0  B7 00                                   mov bh,00h
  268   01E2  B9 0000                                 mov cx,0
  269   01E5  B6 48                                   mov dh,48h
  270   01E7  B2 79                                   mov dl,79h
  271   01E9  CD 10                                   int 10h
  272                                                 ;ends clearing screen
  273   01EB  B8 0013                                 mov ax,0013h
  274   01EE  CD 10                                   int 10h
  275
  276   01F0  8A 0E 0108r                             mov cl,num
  277   01F4  BD 0111r                                mov bp,offset n
  278   01F7  BB 0125r                                mov bx,offset linklist
  279   01FA  83 C5 02                      copy_ll:add bp,02h
  280   01FD  3E: 8A 46 00                            mov al,ds:[bp]
  281   0201  88 07                                   mov [bx],al
  282   0203  43                                      inc bx
  283   0204  C6 07 2D                                mov byte ptr [bx],45
  284   0207  43                                      inc bx
  285   0208  C6 07 3E                                mov byte ptr [bx],62
  286   020B  43                                      inc bx
  287   020C  83 C5 03                                add bp,03h
  288   020F  FE C9                                   dec cl
  289
  290   0211  75 E7                                   jnz copy_ll
  291
  292   0213  8A 0E 0108r                             mov cl,num
  293   0217  FE C9                                   dec cl
  294   0219  B0 03                                   mov al,03h
  295   021B  F6 E1                                   mul cl
```

```
296   021D  FE C0                        inc al
297   021F  A2 0143r                     mov pointerl,al
298
299   0222  B0 01                        mov al,01h
300   0224  B7 00                        mov bh,0h
301   0226  B3 0A                        mov bl, 0ah
302   0228  8A 0E 0143r                  mov cl,pointerl
303   022C  B5 00                        mov ch,00h
304   022E  B2 04                        mov dl,4h
305   0230  B6 04                        mov dh,4h
306   0232  BD 0125r                     mov bp,offset linklist
307   0235  B4 13                        mov ah,13h
308   0237  CD 10                        int 10h
309
310   0239  B4 07                        mov ah,07h
311   023B  CD 21                        int 21h
312
313   023D  B8 0003                      mov ax,0003h
314   0240  CD 10                        int 10h
315
316   0242  E9 FDD1                      jmp menu
317
318   0245                       code ends
319                              end start
```



HEAP:

A heap is a partially sorted binary tree. Although a heap is not completely in order, it conforms to a sorting principle: every node has a value less than either of its children. Additionally, a heap is a "complete tree" -- a complete tree is one in which there are no gaps between leaves. A heap is a specialized tree-based data structure that satisfied the heap property: if B is a child node of A, then key(A) ≥ key(B). This implies that an element with the greatest key is always in the root node, and so such a heap is sometimes called a max-heap. Of course, there's also a minheap.



CODE FOR HEAP:

maxheap.asm

```
 1    0000                          data segment
 2    0000  0A 0D 43 48 4F 4F53+  str1 db 0ah,0dh,"CHOOSE OPERATION",0ah,0dh,'$'
 3          45 20 4F 50 45 52     41+
 4          54 49 4F 4E 0A 0D     24
 5    0015  0A 0D 50 20 46 4F           52+  str2 db 0ah,0dh,"P FOR INSERT",0AH,0DH,'$'
 6          20 49 4E 53 45 52     54+
 7          0A 0D 24
 8    0026  0A 0D 52 20 46 4F           52+  str3 db 0ah,0dh,"R FOR DELETING",0ah,0dh,'$'
 9          20 44 45 4C 45 54     49+
10          4E 47 0A 0D 24
11    0039  0A 0D 44 20 46 4F           52+  str8 db 0ah,0dh,"D FOR DISPLAY",0ah,0dh,'$'
12          20 44 49 53 50 4C     41+
```

```
   13          59 0A 0D 24
   14    004B  0A 0D 41 6E 79 20          6B+  str9 db 0ah,0dh,"Any key for exit",0ah,0dh,'$'
   15          65 79 20 66 6F 72     20+
   16          65 78 69 74 0A 0D     24
   17    0060  0A 0D 45 6E 74 65          72+  str4 db 0ah,0dh,"Enter value to be  pushed",0ah,0dh,'$'
   18          20 76 61 6C 75 65     20+
   19          74 6F 20 62 65 20     70+
   20          75 73 68 65 64 0A     0D+
   21          24
   22    007D  0A 0D 4F 76 65 72          66+  str5 db 0ah,0dh,"Overflow",0ah,0dh,'$'
   23          6C 6F 77 0A 0D 24
   24    008A  0A 0D 44 6F 20 79          6F+  str6 db 0ah,0dh,"Do you want to continue?
(y/n)",0ah,0dh,'$'
   25          75 20 77 61 6E 74     20+
   26          74 6F 20 63 6F 6E     74+
   27          69 6E 75 65 3F 20     28+
   28          79 2F 6E 29 0A 0D     24
   29    00AD  0A 0D 45 6D 70 74          79+  str7 db 0ah,0dh,"Empty",0ah,0dh,'$'
   30          0A 0D 24
   31    00B7  0A 0D 50 6F 70 70          69+  str10 db 0ah,0dh,"Popping ",0ah,0dh,'$'
   32          6E 67 20 20 0A 0D     24
   33    00C5  01*(00)                 choose db 1 dup(0)
   34    00C6  01*(00)                 max db 1 dup(0)
   35    00C7  01*(00)                 num db 1 dup(0)
   36    00C8  01*(00)                 ele db 1 dup(0)
   37    00C9  01*(00)                 i db 1 dup(0)
   38    00CA  01*(00)                 j db 1 dup(0)
   39    00CB  01*(00)                 t db 1 dup(0)
   40    00CC  01*(00)                 x db 1 dup(0)
   41    00CD  01*(00)                 k db 1 dup(0)
   42    00CE  64*(00)                 array db 100 dup(0)
   43    0132                          data ends
   44    0000                          code segment
   45                                        assume cs:code, ds:data
   46    0000  B8 0000s                      start: mov     ax,data
   47    0003  8E D8                                mov   ds,ax
   48    0005  C6 06 00C6r 0A                       mov   max,10
   49    000A  C6 06 00C7r 00                       mov   num,00
   50    000F  C6 06 00C8r 00                       mov   ele,00
   51    0014  BA 0000r                      menu:  mov   dx, offset str1
   52    0017  B4 09                                mov   ah,09h
   53    0019  CD 21                                int     21h
   54    001B  BA 0015r                             mov   dx, offset str2
   55    001E  B4 09                                mov   ah,09h
   56    0020  CD 21                                int     21h
   57    0022  BA 0026r                             mov   dx, offset str3
   58    0025  B4 09                                mov   ah,09h
   59    0027  CD 21                                int     21h
```

```
60   0029  BA 0039r                    mov  dx, offset str8
61   002C  B4 09                       mov  ah,09h
62   002E  CD 21                       int  21h
63   0030  BA 004Br                    mov  dx, offset str9
64   0033  B4 09                       mov  ah,09h
65   0035  CD 21                       int  21h
66   0037  B4 07                       mov  ah,07
67   0039  CD 21                       int  21h
68   003B  3C 70                       cmp  al, 'p'
69   003D  74 0B                       je p
70   003F  3C 72                       cmp  al, 'r'
71   0041  74 66                       je call_r
72   0043  3C 64                       cmp  al, 'd'
73   0045  74 67                       je call_d
74   0047  E9 018D                     jmp  e
75   004A  A0 00C7r             p:     mov  al, num
76   004D  8A 16 00C6r                 mov  dl, max
77   0051  3A C2                       cmp  al,dl
78   0053  75 09                       jne  x2
79                                     ; overflow
80   0055  BA 007Dr                    mov  dx,offset str5
81   0058  B4 09                       mov  ah,09
82   005A  CD 21                       int  21h
83   005C  EB B6                       jmp  menu
84   005E                       x2:    ;ask for value to be entered
85   005E  BA 0060r                    mov  dx,offset str4
86   0061  B4 09                       mov  ah,09
87   0063  CD 21                       int  21h
88   0065  B4 01                       mov  ah,01
89   0067  CD 21                       int  21h
90   0069  A2 00C8r                    mov  ele,al
91   006C  A0 00C7r                    mov  al,num
92   006F  A2 00C9r                    mov  i, al
93   0072  FE 06 00C9r                 inc  i                  ;i=size+1
94   0076  80 3E 00C9r 01       x11:   cmp  i, 01              ;    check i!=1
95   007B  74 34                       je x1
96   007D  A0 00C9r                    mov  al, i
97   0080  B4 00                       mov  ah, 00
98   0082  B2 02                       mov  dl, 02
99   0084  F6 F2                       div  dl
100  0086  8A D8                       mov  bl,al              ;bl=i/2
101  0088  B7 00                       mov  bh,00
102  008A  A0 00C8r                    mov  al, ele
103  008D  38 80 00CEr                 cmp  [si+array+bx],al   ;check heap[i/2]<ele
104  0091  73 1E                       jnc  x1
105  0093  8A 80 00CEr                 mov  al,[si+array+bx]
106  0097  53                          push bx
107  0098  8A 1E 00C9r                 mov  bl, i              ;    bl=i
```

```
  108   009C  B7 00                          mov  bh, 00
  109   009E  88 80 00CEr                    mov  [si+array+bx],al     ;heap[i]=heap[i/2]
  110   00A2  5B                             pop  bx
  111   00A3  88 1E 00C9r                    mov  i,bl                 ;i=i/2
  112   00A7  EB CD                          jmp  x11
  113   00A9  EB 29 90                 call_r: jmp  r
  114   00AC  EB 9C                    call_p: jmp  p
  115   00AE  E9 00C8                  call_d: jmp  call_d2
  116   00B1  8A 1E 00C9r              x1:     mov  bl,i
  117   00B5  B7 00                          mov  bh,00
  118   00B7  A0 00C8r                       mov  al,ele
  119   00BA  88 80 00CEr                    mov[si+array+bx],al       ;heap[i]=ele
  120   00BE  FE 06 00C7r                    inc  num
  121   00C2  BA 008Ar                       mov  dx, offset str6
  122   00C5  B4 09                          mov  ah,09
  123   00C7  CD 21                          int  21h
  124   00C9  B4 01                          mov  ah,01
  125   00CB  CD 21                          int  21h
  126   00CD  3C 79                          cmp  al, 'y'
  127   00CF  74 DB                          je call_p
  128   00D1  E9 FF40                        jmp  menu
  129
  130   00D4  80 3E 00C7r 00           r:      cmp  num, 00
  131   00D9  75 0A                          jne  y
  132   00DB  BA 00ADr                       mov  dx, offset str7      ;empty heap
  133   00DE  B4 09                          mov  ah,09
  134   00E0  CD 21                          int  21h
  135   00E2  E9 FF2F                        jmp  menu
  136   00E5                           y:      ; pop max ele
  137   00E5  BA 00B7r                       mov  dx,offset str10
  138   00E8  B4 09                          mov  ah,09
  139   00EA  CD 21                          int  21h
  140   00EC  8A 84 00CFr                    mov  al,[si+array+1]
  141   00F0  A2 00CCr                       mov  x,al                 ;x=heap[1] ie ele      to be
deleted
  142   00F3  8A D0                          mov  dl,al
  143   00F5  B4 02                          mov  ah,2
  144   00F7  CD 21                          int  21h
  145   00F9  8A 1E 00C7r                    mov  bl,num
  146   00FD  B7 00                          mov  bh,00
  147   00FF  8A 80 00CEr                    mov  al,[si+array+bx]
  148   0103  A2 00CDr                       mov  k,al                 ;k= heap[n]
  149   0106  FE 0E 00C7r                    dec  num                  ;dec size      of heap
  150   010A  A0 00C7r                       mov  al,num
  151   010D  C6 06 00C9r 01                 mov  i, 01
  152   0112  C6 06 00CAr 02                 mov  j, 02
  153   0117  A0 00C7r                 y2:     mov  al, num
  154   011A  FE C0                          inc  al
```

```
155   011C  38 06 00CAr                cmp  j,al                ;check j<=n
156   0120  73 5A                      jnc  y1
157   0122  A0 00C7r                   mov  al,num
158   0125  38 06 00CAr                cmp  j,al                ;check j<n
159   0129  73 1E                      jnc  y11
160   012B  8A 1E 00CAr                mov  bl,j
161   012F  B7 00                      mov  bh,00               ;bx=j
162   0131  8A 80 00CEr                mov  al,[si+array+bx]
163   0135  8A 1E 00CAr                mov  bl,j
164   0139  FE C3                      inc  bl
165   013B  B7 00                      mov  bh,00               ;bx=j+1
166   013D  8A 90 00CEr                mov  dl,[si+array+bx]
167
168   0141  3A C2                      cmp  al,dl               ;     check heap[j]<heap[j+1]
169   0143  73 04                      jnc  y11
170   0145  FE 06 00CAr                inc  j
171   0149                        y11:
172   0149  A0 00CDr                   mov  al,k
173   014C  8A 1E 00CAr                mov  bl,j
174   0150  B7 00                      mov  bh,00
175   0152  8A 90 00CEr                mov  dl,[si+array+bx]    ;dl=heap[j]
176   0156  3A C2                      cmp  al,dl               ;     check k>=heap[j] if yes,
break
177   0158  73 22                      jnc  y1
178   015A  8A 1E 00C9r                mov  bl,i
179   015E  B7 00                      mov  bh,00
180   0160  88 90 00CEr                mov  [si+array+bx],dl    ;heap[i]=heap[j]
181   0164  A0 00CAr                   mov  al,j
182   0167  A2 00C9r                   mov  i,al                ;i=j
183   016A  A0 00CAr                   mov  al,j
184   016D  B1 02                      mov  cl,2
185   016F  F6 E1                      mul  cl
186   0171  A2 00CAr                   mov  j, al               ;j=jx2
187   0174  EB A1                      jmp  y2
188   0176  E9 FF5B               call_r2:jmp  r
189   0179  EB 20 90              call_d2:jmp  d
190   017C  A0 00CDr              y1:      mov  al,k
191   017F  8A 1E 00C9r                mov  bl,i
192   0183  B7 00                      mov  bh,00
193   0185  88 80 00CEr                mov  [si+array+bx],al    ;heap[i]=k
194   0189  BA 008Ar                   mov  dx, offset str6
195   018C  B4 09                      mov  ah,09
196   018E  CD 21                      int  21h
197   0190  B4 01                      mov  ah,01
198   0192  CD 21                      int  21h
199   0194  3C 79                      cmp  al, 'y'
200   0196  74 DE                      je call_r2
201   0198  E9 FE79                    jmp  menu
```

```
202   019B  80 3E 00C7r 00          d:      cmp   num, 00
203   01A0  75 0A                           jne   z
204   01A2  BA 00ADr                        mov   dx, offset str7       ;empty heap
205   01A5  B4 09                           mov   ah,09
206   01A7  CD 21                           int   21h
207   01A9  E9 FE68                         jmp   menu
208   01AC                          z:
209   01AC  B3 01                           mov   bl, 01
210   01AE  B7 00                           mov   bh, 00
211   01B0  C6 06 00CBr 01                  mov   t,01
212   01B5  8A 1E 00CBr             z1:     mov   bl,t
213   01B9  B7 00                           mov   bh,00
214   01BB  8A 90 00CEr                     mov   dl,[si+array+bx]
215   01BF  B4 02                           mov   ah,02
216   01C1  CD 21                           int   21h
217   01C3  FE 06 00CBr                     inc   t
218   01C7  A0 00CBr                        mov   al,t
219   01CA  8A 16 00C7r                     mov   dl,num
220   01CE  FE C2                           inc   dl
221   01D0  3A C2                           cmp   al,dl
222   01D2  75 E1                           jne   z1
223   01D4  E9 FE3D                         jmp   menu
224   01D7  CC                      e:      int   3
225
226   01D8                    code ends
227                           end start
```