Bài tập cơ sở lập trình
Pointer

Họ và tên : Bùi Chí Giang
MSSV : 25120053

Câu 2.1 :
print out :
0x16f7bf368 : địa chỉ của a vì con trỏ b đang lưu địa chỉ của a
3 : giá trị của a vì đang giải tham chiếu con trỏ b
0x16f7bf360 : địa chỉ của con trỏ b vì đang tham chiếu đến địa chỉ của con trỏ b
3 : giá trị của a
0x16f7bf368 : địa chỉ của a vì đang tham chiếu đến địa chỉ của a
câu 2.2
print out :
100 : giá trị x vì ip2 lưu địa chỉ của x và không đổi khi dừng chương trình
1.2 : giá trị y vì fp lưu địa chỉ của y và gán *fp = 1.2 nên giá trị của y cũng thay đổi là 1.2
400 : giá trị z vì ip1 lưu địa chỉ của z và gán *ip1= *ip2 (x) + 300 = 400 nên giá trị z là 400
0x16aeff364 : địa chỉ của z vì ip1 đang trỏ đến z (đang lưu địa chỉ của z)
400 : giải tham chiếu ip1 chính là giá trị của z = 400
0x16aeff348 : địa chỉ của ip1 vì đang tham chiếu đến địa chỉ của con trỏ ip1
0x16aeff368 : địa chỉ của x vì ip2 đang trỏ đến x (đang lưu địa chỉ của x)
100 : giá trị của x vì đang giải tham chiếu ip2 chính là giá trị của x = 100
0x16aeff340 : địa của ip2 vì đang tham chiếu đến địa chỉ của con trỏ ip2
0x16aeff360 : địa chỉ của y vì fp đang trỏ đến y (đang lưu địa chỉ của y)
1.2 : giá trị của y vì đang giải tham chiếu fp chính là giá trị y = 1.2
0x16aeff338 :địa chỉ của fp vì đang tham thiếu đến địa chỉ con trỏ fp
Z���?� : địa chỉ của ch vì chp đang trỏ đến ch (đang lưu địa chỉ của ch)
Z : giá trị của ch vì đang giải tham chiếu con trỏ chp
0x16aeff350 : địa chỉ của con trỏ chp vì đang tham chiếu đến địa chỉ con trỏ chp
câu 2.3 :
print out :
2
2
bị lỗi double free do 2 con trỏ a b cùng trỏ vào cùng 1 vùng nhớ
khi giải phóng a và tiếp tục giải phóng b tức là giải phóng tiếp vùng nhớ đã giải phóng làm gây ra lỗi double free làm crash chương trình
câu 2.4
print out :
3
5
problem :
cấp phát vùng nhớ cho con trỏ p nhưng không có lệnh “ delete p “ , dẫn đến memory leak

câu 2.5
*p = 50 : ở dòng 10 , s = p tức là s trỏ tới vùng nhớ mà p đang trỏ đến , *s=50 => *p=50
q=8 : ở dòng 9 , q = v = 8 => q =8
*r = 8 : ở dòng 5 , r đang trỏ tới vùng nhớ của p mà p đang trỏ tới q , dòng 9 q =v =8 => *r=8
v= 8 : không đỗi
*s = 50 : ở dòng 11 , *s=50
câu 2.6
dòng 1 : khởi tạo con trỏ p , q , biến v , mảng nom 5 phần tử
dòng 2 : p trỏ đến v (p lưu địa chỉ v)
dòng 3 : giá trị con trỏ p = 12 => v = 12
dòng 4 : q trỏ đến vùng nhớ của q đang trỏ đến tức là địa chỉ của v => *q = 12
dòng 5 : giá trị nom[0] lưu giá trị *p = 12
dòng 6 : p trỏ đến địa chỉ của nom[0]
dòng 7 : p trỏ đến địa chỉ của nom[1]
dòng 8 : giá trị nom[2] =12
dòng 9 : gán giá trị *p =13 => nom[1] = 13
dòng 10 : gán giá trị *q = 10 => v = 10
dòng 11 : v = 11 => *q = 11
dòng 12 : giá trị của nom[4]=16
dòng 13 : p trỏ đến nom[3]
dòng 14 : giá trị *p=10 => nom[3]=10
dòng 15 : p trỏ đến nom[2]
result :
*p =12 , *q = 11 , v = 11 , nom[5]= { 12 , 13 , 12 , 10 , 16 }
câu 2.7
C . No error
vì con trỏ x được khai báo nhưng chưa khởi tạo nên vẫn đúng về mặt syntax
câu 2.8
D. lice ice ce e
vì khởi tạo con trỏ x trỏ chứa giá trị là “Alice” và lưu địa chỉ của x[0] , *x = x[5] (strlen = 5)
=> x[0]=x[5] = 1 giá trị rác vì x[5] chưa được khai báo , khi in chỉ in được từ lice - > e
câu 2.9
D. ce
vì s++ + 3 sẽ trỏ đến vị trí c sau đó in ra ce
câu 2.10
B. 2,15,6,8,10
vì hàm change sẽ thay a[1]=a[4]+5=15
câu 2.11
B. 20,4,4
vì sizeof(arr) = 4*5 = 20 (5 phần tử)
sizeof(*arr)=sizeof(arr[0])=4

câu 2.12
D.300
vì str lưu địa chỉ str[0] . str++ trỏ đến ‘d’ , str++ trỏ đến ‘\’ , str -2 trở về ‘%’ nên cú pháp printf trở thành printf{“%d\n”,300) => output là 300
câu 2.13
A.* dùng để giải tham chiếu
câu 2.14
A.x is a pointer to a string , y is a string
câu 2.15
D. point to a tye
point không thể trỏ đến 1 kiểu dữ liệu
câu 2.16
C. int i ; double*dp=&i
vì i là int mà dp có type là double
câu 2.17
B.p now points to b
vì p = q tức là p trỏ đến vùng nhớ mà q đang trỏ đến tức là b
câu 2.18
D . random number
vì arr[1] đang chứa địa chỉ của b
câu 2.19
A.ABCDEFGHIJ
vì *(arr + i) = “\0” nằm ngoài vòng lặp nên không ảnh hưởng do biến i chỉ có tác dụng trong vòng lặp
câu 2.20
A, fg
vì ptr+5 sẽ trỏ đến index 5 của f nên in ra fg
câu 2.21
D. all of them
câu 2.22
D.all of them
vì ở câu a : int * p ; câu b : const int *p ; câu c : int * const p
câu 2.23
B.const
câu 2.24
C . the new operator
câu 2.25
B. indirection (giải tham chiếu)
câu 2.26
A.sizeof
câu 2.27
A.pointer contains an address of a variable
câu 2.28
C.3 (0 , NULL , address)

câu 2.29
C.Address operator (&)
câu 2.30
D. 129,A
vì gán tham chiếu cho = ch = ‘A’ , cho+= a = 64 + 32 = 96 . *ptr = a = 96+32=129
câu 2.31
D.complie error
vì con trỏ hằng nên (*ptr)++ bị lỗi
câu 2.32
B. 10,20,30,40,50,
câu 2.33
C.14
vì arr lưu địa chỉ arr[0] nên *arr + 10 = 4 +10 = 14
câu 2.34
a.10
vì ra tham chiếu đến địa chỉ của a nên in ra ra chính là in ra a
câu 2.35
output : 2 vì ptr = a chính là lưu địa chỉ a[0] nên *(ptr +1) = a[1] = 2
câu 2.36
output : 15 vì con trỏ ptr trỏ đến vùng nhớ của biến a nên *ptr = a = *ptr*3= 5*3=15
câu 2.27
output : 222 vì con trỏ j trỏ đến vùng nhớ của biến i nên phép tính = 6*6*6 + 6 = 222
câu 2.38
output : x = 21 , y=504 , z = 504
vì x++ nên x=21 , *y++ = *(y++) tức là y nhảy sang ô kế tiếp hay địa chỉ mà y trỏ đến tăng lên 4 byte = 500 + 4 = 504 , tương tự với z
câu 2.39
output : x = 10 , y = 10 , z =10
vì y tham chiếu đến x , z tham chiếu đến y nên khi giải tham chiếu y sẽ ra x = 10 , khi giải tham chiếu **z cũng sẽ ra x = 10
câu 2.40
D.C A
vì *ppp++ tức là *(ppp++) do đó con trỏ ppp đang chỏ đến ‘B’ , nên khi thực hiện *++ppp tức là giải tham chiếu giá trị tiếp theo ‘C’ , –*ppp tức là giải tham chiếu ra ‘B’ sau đó trừ đi 1 tức là ‘A’
câu 2.41
B,runtime error
do con trỏ ptr đang trỏ vào 1 địa chỉ rác nên sẽ dẫn đến Segmentation Fault
câu 2.42
C. address , address
vì giải tham chiếu &ptr chính là ptr chứa địa chỉ của a còn tham chiếu đến *ptr chính là a cũng chính là địa chỉ của a
câu 2.43
D. vì num++ chỉ tăng giá trị không làm thay đỗi địa chỉ của num

câu 2.44
A.address address value
vì k chứa địa chỉ của j , *k chứa địa chỉ của i , **k là giá trị của i
câu 2.45
A. vì y trỏ đến x , z trỏ đến vùng nhớ mà y đang trỏ chính là x
câu 2.46
B. track[0]=*striker -=10 = 10 -10 =0
câu 2.47
D,vì gán tham chiếu cho = ch = ‘A’ , cho+= a = 64 + 32 = 96 . *ptr = a = 96+32=129
câu 2.48
C,compilation error vì con trỏ hằng không thể thay đỗi (*ptr)++
câu 2.49
C. BBB
vì pp trỏ tới sptr , ++pp tức là sptr[1] , **++pp tức là giải tham thiếu sptr[2] = str +1 = BBBBB mà +2 tức là bỏ đi 2 BB đầu chỉ còn BBB
câu 2.50
A.5 4 3 2 1
vì đây là hàm hoán đỗi vị trí đầu với cuối , thứ 2 với gần cuối