8/26/2011

C

C PROGRAMMING QUESTIONS AND ANSWER

http://cquestionbank.blogspot.com | Ritesh kumar

(1) What will be output if you will compile and execute the following c code?

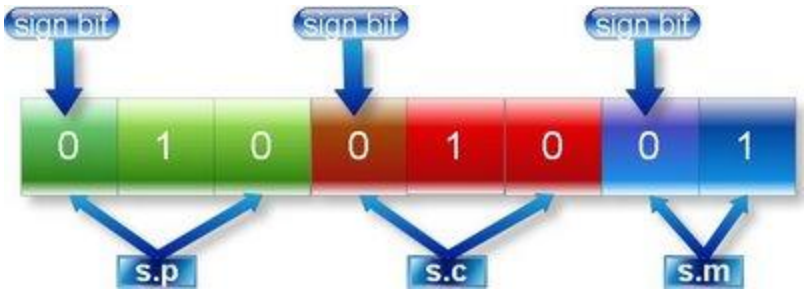
```
#include<stdio.h>

struct marks{
    int p:3;
    int c:3;
    int m:2;
};

int main(){
    struct marks s={2,-6,5};
    printf("%d %d %d",s.p,s.c,s.m);
    return 0;

}
```

(a) 2 -6 5
(b) 2 -6 1
(c) 2 2 1
(d) Compiler error
(e) None of these

Answer: (c)
Explanation:

Binary value of 2: 00000010 (Select three two bit)
Binary value of 6: 00000110
Binary value of -6: 11111001+1=11111010
(Select last three bit)
Binary value of 5: 00000101 (Select last two bit)

Complete memory representation:



(2) What will be output if you will compile and execute the following c code?

```
#include<stdio.h>

int main(){
     int huge*p=(int huge*)0XC0563331;
     int huge*q=(int huge*)0xC2551341;
     *p=200;
     printf("%d",*q);
     return 0;

}
```

(a)0
(b)Garbage value
(c)null
(d) 200
(e)Compiler error

Answer: (d)
Explanation:

Physical address of huge pointer p
Huge address: 0XC0563331
Offset address: 0x3331
Segment address: 0XC056

Physical address= Segment address * 0X10 + Offset address
=0XC056 * 0X10 +0X3331
=0XC0560 + 0X3331
=0XC3891
Physical address of huge pointer q
Huge address: 0XC2551341
Offset address: 0x1341
Segment address: 0XC255
Physical address= Segment address * 0X10 + Offset address
=0XC255 * 0X10 +0X1341
=0XC2550 + 0X1341
=0XC3891
Since both huge pointers p and q are pointing same physical address so content of q will also same as content of q.

(3) Write c program which display mouse pointer and position of pointer. (In x coordinate, y coordinate)?

Answer:

```
#include<dos.h>
#include<stdio.h>

int main(){
    union REGS i,o;
    int x,y,k;
    //show mouse pointer
    i.x.ax=1;
    int86(0x33,&i,&o);
    while(!kbhit()) //its value will false when we hit key in the key board
    {
```

```
        i.x.ax=3; //get mouse position
        x=o.x.cx;
        y=o.x.dx;
        printf("(%d , %d)",x,y);
        delay(250);
        int86(0x33,&i,&o);
    }
    return 0;
}
```

(4) Write a c program to create dos command: dir.

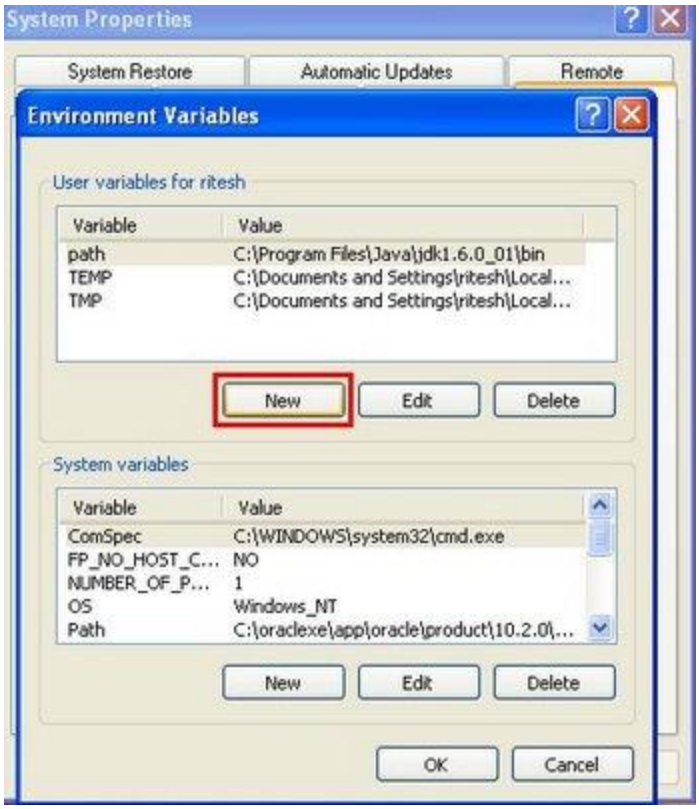
Answer:

Step 1: Write following code.

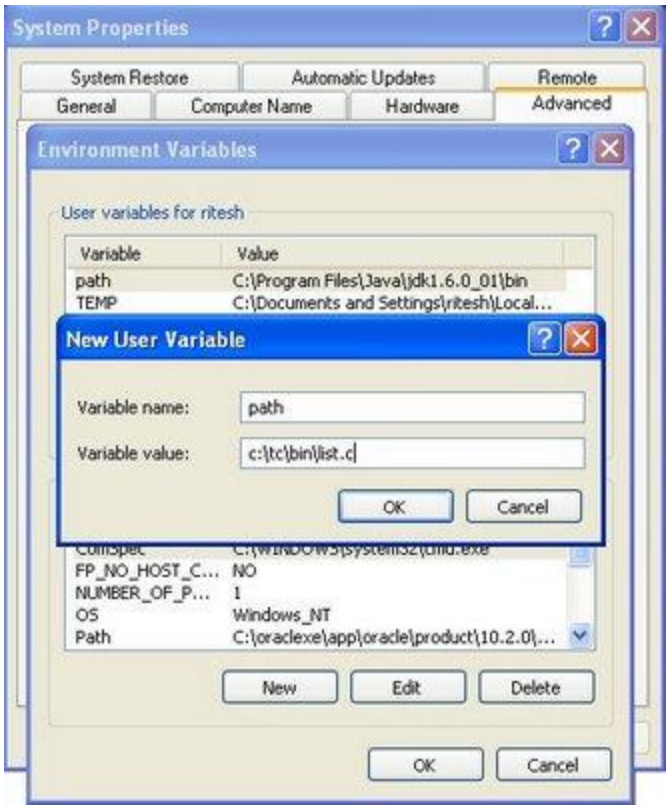
```
#include <stdio.h>
#include <dos.h>

int main(int count,char *argv[]){
    struct find_t q ;
    int a;
    if(count==1)
        argv[1]="*.*";
        a = _dos_findfirst(argv[1],1,&q);
        if(a==0){
            while (!a){
                printf(" %s\n", q.name);
                a = _dos_findnext(&q);
            }
        }
        else{
            printf("File not found");
    }
    return 0;
```

```
}
```

Step 2: Save the as list.c (You can give any name)
Step 3: Compile and execute the file.
Step 4: Write click on My computer of Window XP operating system and select properties.
Step 5: Select Advanced -> Environment Variables
Step 6: You will find following window:
Click on new button (Button inside the red box)



Step 7: Write following:
Variable name: path
Variable value: c:\tc\bin\list.c (Path where you have saved)



Step 8: Open command prompt and write list and press enter.

(5) What will be output if you will compile and execute the following c code?

```
#include<stdio.h>

int main(){
     int i;
     float a=5.2;
     char *ptr;
     ptr=(char *)&a;
     for(i=0;i<=3;i++)
          printf("%d ",*ptr++);
     return 0;
}
```

(a)0 0 0 0
(b)Garbage Garbage Garbage Garbage
(c)102 56 -80 32
(d)102 102 -90 64
(e)Compiler error

Answer: (d)
Explanation:

In c float data type is four byte data type while char pointer ptr can point one byte of memory at a time.
Memory representation of float a=5.2

ptr pointer will point first fourth byte then third byte then second byte then first byte.

Content of fourth byte:
Binary value=01100110
Decimal value= 64+32+4+2=102

Content of third byte:
Binary value=01100110
Decimal value=64+32+4+2=102
Content of second byte:

```
Binary value=10100110
Decimal value=-128+32+4+2=-90

Content of first byte:
Binary value=01000000
Decimal value=64
```

Note: Character pointer treats MSB bit of each byte i.e. left most bit of above figure as sign bit.

(6) What will be output if you will compile and execute the following c code?

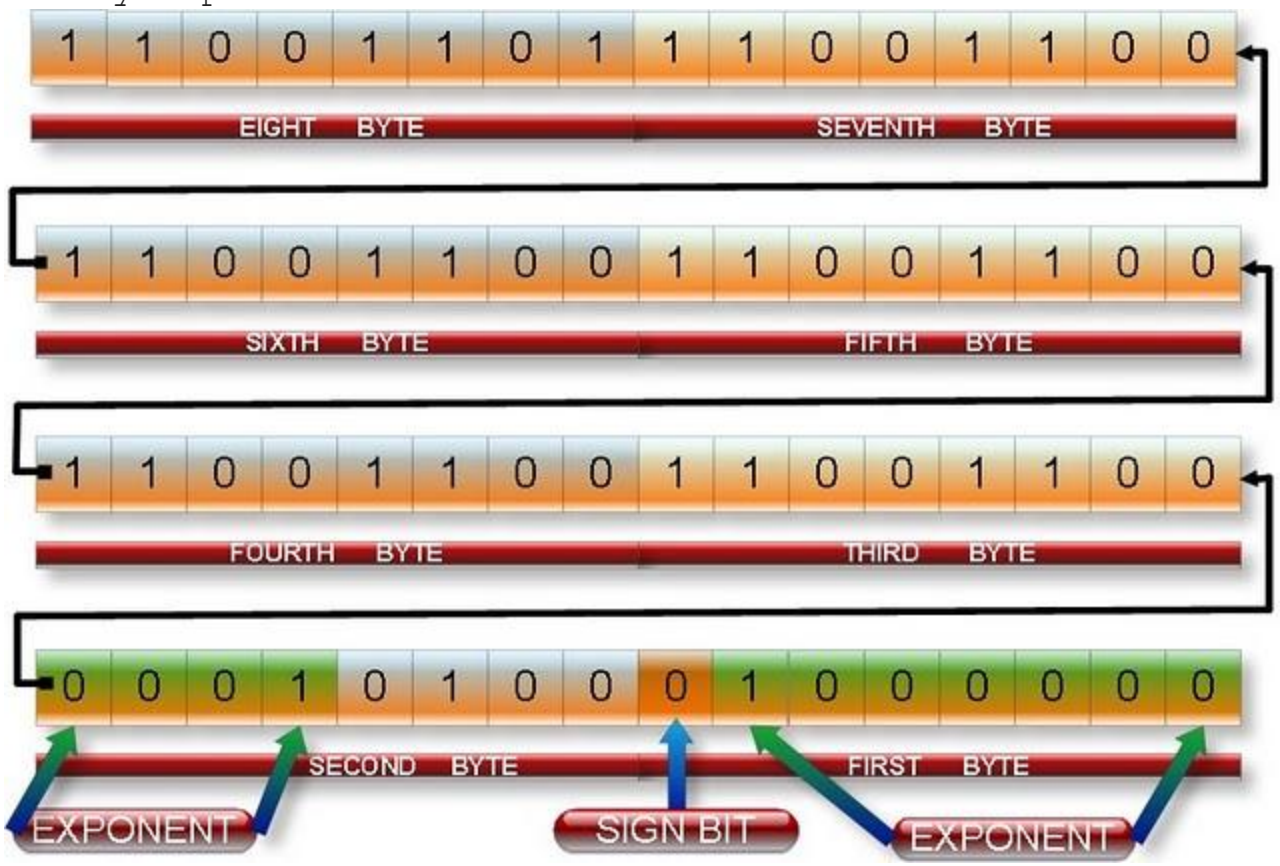
```
#include<stdio.h>

int main(){
    int i;
    double a=5.2;
    char *ptr;
    ptr=(char *)&a;
    for(i=0;i<=7;i++)
        printf("%d ",*ptr++);
    return 0;

}
```

(a) -51 -52 -52 -52 -52 -52 20 64
(b) 51 52 52 52 52 52 20 64
(c) Eight garbage values.
(d) Compiler error
(e) None of these

Answer: (a)
Explanation:

In c double data type is eight byte data type while char pointer ptr can point one byte of memory at a time.

Memory representation of double a=5.2



ptr pointer will point first eighth byte then seventh byte then sixth byte then fifth byte then fourth byte then third byte then second byte then first byte as shown in above figure.

Content of eighth byte:
Binary value=11001101
Decimal value= -128+64+8+4+1=-51

Content of seventh byte:

```
Binary value=11001100
Decimal value= -128+64+8+4=-52

Content of sixth byte:
Binary value=11001100
Decimal value= -128+64+8+4=-52

Content of fifth byte:
Binary value=11001100
Decimal value= -128+64+8+4=-52

Content of fourth byte:
Binary value=11001100
Decimal value= -128+64+8+4=-52

Content of third byte:
Binary value=11001100
Decimal value= -128+64+8+4=-52

Content of second byte:
Binary value=000010100
Decimal value=16+4=20
Content of first byte:
Binary value=01000000
Decimal value=64
```

Note: Character pointer treats MSB bit of each byte i.e. left most bit of above figure as sign bit.

(7) What will be output if you will compile and execute the following c code?

```
#include<stdio.h>

int main(){
```

```
    printf("%s","c" "question" "bank");
    return 0;
}
```

(a) c question bank
(b) c
(c) bank
(d) cquestionbank
(e) Compiler error

Answer: (d)
Explanation:

In c string constant “xy” is same as “x” “y”

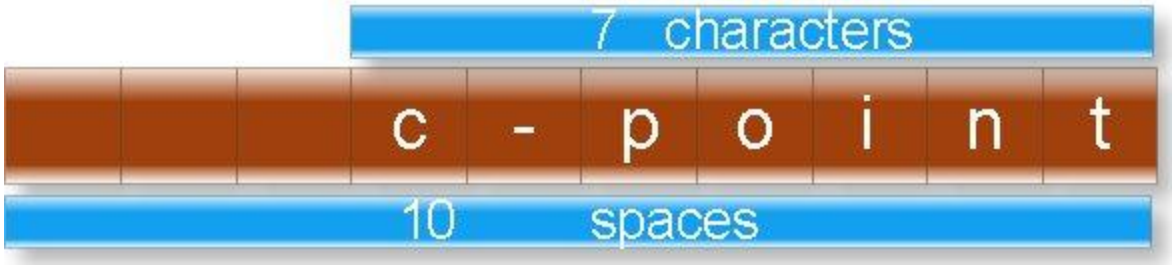
(8) What will be output if you will compile and execute the following c code?

```
#include<stdio.h>

int main(){
    char *str="c-pointer";
    printf("%*.*s",10,7,str);
    return 0;
}
```

(a) c-pointer
(b) c-pointer
(c) c-point
(d) cpointer null null
(e) c-point

Answer: (e)
Explanation:

Meaning of %*.*s in the printf function:
First * indicates the width i.e. how many spaces will take to print the string and second * indicates how many characters will print of any string.
Following figure illustrates output of above code:



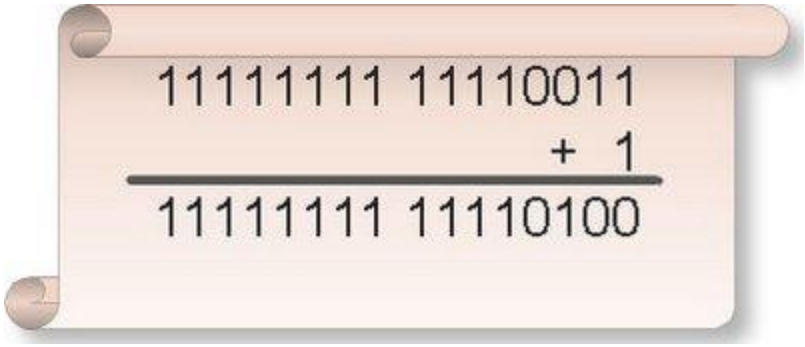
(9) What will be output if you will compile and execute the following c code?

```
#include<stdio.h>

int main(){
    int a=-12;
    a=a>>3;
    printf("%d",a);
    return 0;
}
```

(a) -4
(b) -3
(c) -2
(d) -96
(e) Compiler error

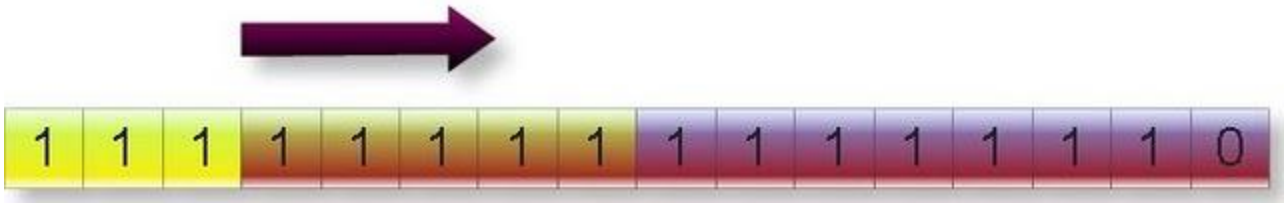
Answer :(c)
Explanation:

Binary value of 12 is: 00000000 00001100
Binary value of -12 wills 2's complement of 12 i.e.

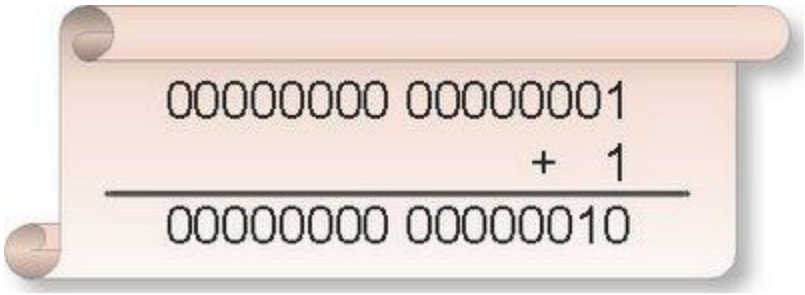


So binary value of -12 is: 11111111 11110100

Right shifting rule:
Rule 1: If number is positive the fill vacant spaces in the left side by 0.
Rule 2: If number is negative the fill vacant spaces in the left side by 1.
In this case number is negative. So right shift all the binary digits by three space and fill vacant space by 1 as shown following figure:



Since it is negative number so output will also a negative number but its 2's complement.



Hence final output will be:

And its decimal value is: 2
Hence output will be:-2

(10) What will be output if you will compile and execute the following c code?

```
#include<stdio.h>
#include <string.h>

int main(){
    printf("%d %d",sizeof("string"),strlen("string"));
    return 0;
}
```

(a) 6 6
(b) 7 7
(c) 6 7
(d) 7 6
(e) None of these

Answer: (d)
Explanation:

Sizeof operator returns the size of string including null character while strlen function returns length of a string excluding null character.

(11) What will be output if you will compile and execute the following c code?

```
#include<stdio.h>

int main(){
     static main;
     int x;
     x=call(main);
     printf("%d ",x);
     return 0;
}

int call(int address){
     address++;
     return address;
}
```

(a) 0
(b) 1
(c) Garbage value
(d) Compiler error
(e) None of these

Answer: (b)
Explanation:

As we know main is not keyword of c but is special type of function. Word main can be name variable in the main and other functions.

(12) What will be output if you will compile and execute the following c code?

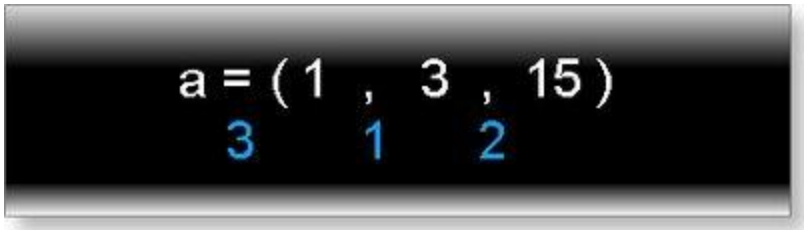
```
#include<stdio.h>

int main(){
     int a,b;
     a=1,3,15;
     b=(2,4,6);
     printf("%d ",a+b);
     return 0;
}
```

(a) 3
(b) 21
(c) 17
(d) 7
(e) Compiler error

Answer: (d)
Explanation:

In c comma behaves as separator as well as operator.
a=1, 3, 15;
b= (2, 4, 6);
In the above two statements comma is working as operator. Comma enjoys least precedence and associative is left to right.
Assigning the priority of each operator in the first statement:

Hence 1 will assign to a.
Assigning the priority of each operator in the second statement:



(13) What will be output if you will compile and execute the following c code?

```
#include<stdio.h>

int extern x;
int main()
    printf("%d",x);
    x=2;
    return 0;
}

int x=23;
```

(a) 0
(b) 2
(c) 23
(d) Compiler error
(e) None of these

Answer: (c)
Explanation:

extern variables can search the declaration of variable anywhere in the program.

(14) What will be output if you will compile and execute the following c code?

```
#include<stdio.h>

int main(){
    int i=0;
    if(i==0){
        i=((5,(i=3)),i=1);
        printf("%d",i);
    }
    else
        printf("equal");
}
```

(a) 5
(b) 3
(c) 1
(d) equal
(e) None of above

Answer: (c)

(15) What will be output if you will compile and execute the following c code?

```
int main(){
    int a=25;
    printf("%o %x",a,a);
    return 0;
}
```

(a) 25 25
(b) 025 0x25

(c) 12 42
(d) 31 19
(e) None of these

Answer: (d)
Explanation:

%o is used to print the number in octal number format.
%x is used to print the number in hexadecimal number format.

Note: In c octal number starts with 0 and hexadecimal number starts with 0x.

(16) What will be output if you will compile and execute the following c code?

```
#include<stdio.h>

#define message "union is\
power of c"
int main(){
    printf("%s",message);
    return 0;
}
```

(a) union is power of c
(b) union ispower of c
(c) union is
Power of c
(d) Compiler error
(e) None of these
Answer: (b)
Explanation:

If you want to write macro constant in new line the end with the character \.

(17) What will be output if you will compile and execute the following c code?

```
#include<stdio.h>

#define call(x) #x
int main(){
    printf("%s",call(c/c++));
    return 0;
}
```

(a)c
(b)c++
(c)#c/c++
(d)c/c++
(e)Compiler error

Answer: (d)
Explanation:

is string operator. It converts the macro function call argument in the string. First see the intermediate file:

```
test.c 1:
test.c 2: void main(){
test.c 3: printf("%s","c/c++");
test.c 4: return 0;
test.c 4: }
test.c 5:
```

It is clear macro call is replaced by its argument in the string format.

(18) What will be output if you will compile and execute the following c code?

```
#include<stdio.h>

int main(){
    if(printf("cquestionbank"))
        printf("I know c");
    else
        printf("I know c++");
    return 0;
}
```

(a) I know c
(b) I know c++
(c) cquestionbankI know c
(d) cquestionbankI know c++
(e) Compiler error

Answer: (c)
Explanation:

Return type of printf function is integer which returns number of character it prints including blank spaces. So printf function inside if condition will return 13. In if condition any non- zero number means true so else part will not execute.

(19) What will be output if you will compile and execute the following c code?

```
#include<stdio.h>
```

```
int main(){
     int i=10;
     static int x=i;
     if(x==i)
          printf("Equal");
     else if(x>i)
          printf("Greater than");
     else
          printf("Less than");
     return 0;

}
```

(a) Equal
(b) Greater than
(c) Less than
(d) Compiler error
(e) None of above

Answer: (d)
Explanation:

Static variables are load time entity while auto variables are run time entity. We cannot initialize any load time variable by the run time variable.
In this example i is run time variable while x is load time variable.

(20) What will be output if you will compile and execute the following c code?

```
#include<stdio.h>

int main(){
```

```
    printf("%s",__DATE__);
    return 0;

}
```

(a) Current system date
(b) Current system date with time
(c) null
(d) Compiler error
(e) None of these

Answer: (a)
Explanation:

__DATE__ is global identifier which returns current system date.

(21) What will be output if you will compile and execute the following c code?

```
#include<stdio.h>

void start();
void end();
#pragma startup start
#pragma exit end
int static i;

int main(){
    printf("\nmain function: %d",++i);
    return 0;

}
```

```
void start(){
    printf("\nstart function: %d",++i);
}

void end(){
    printf("\nend function: %d",++i);
}
```

(a)
main function: 2
start function: 1
end function:3
(b)
start function: 1
main function: 2
end function:3
(c)
main function: 2
end function:3
start function: 1
(d) Compiler error
(e) None of these
Answer: (b)

Explanation:

Every c program start with main function and terminate with null statement. But #pragma startup can call function just before main function and #pragma exit