Poornam Info Vision Technical Test Questions

Q1. What is the output of the following program?

```
main()
{
extern int i;
i=20;
printf(%d,i);
}
```

ANS: Linker Error : Undefined symbol _i

Explanation:

extern storage class in the following declaration,

```
extern int i;
```

specifies to the compiler that the memory for i is allocated in some other program and that address will be given to the current program at the time of linking. But linker finds that no other variable of name i is available in any other program with memory space allocated for it. Hence a linker error has occurred.

Q2. What is the output of the following program?

```
main()
{
printf(%x,-1<<4);
}
```

ANS: fff0

Explanation:

-1 is internally represented as all 1s. When left shifted four times the least significant 4 bits are filled with 0s.The %x format specifier specifies that the integer value be printed as a hexadecimal value.

Q3. What is the output of the following program?

```
main()
{
int i=10;
i=!i>14;
Printf (i=%d,i);
}
```

ANS: i=0

Explanation:

In the expression !i>14 , NOT (!) operator has more precedence than > symbol. ! is a unary logical operator. !i (!10) is 0 (not of true is false). 0>14 is false (zero).

Q4. What is the output of the following program?

```
#include
main()
{
int a[2][2][2] = { {10,2,3,4}, {5,6,7,8} };
int *p,*q;
p=&a[2][2][2];
*q=***a;
printf(%d----%d,*p,*q);
}
```

ANS: SomeGarbageValue---1

Explanation:

p=&a[2][2][2] you declare only two 2D arrays, but you are trying to access the third 2D(which you are not declared) it will print garbage values.

*q=***a starting address of a is assigned integer pointer. Now q is pointing to starting address of a. If you print *q, it will print first element of 3D array.

Q5. What is the output of the following program?

```
main()
{
printf( ab);
printf(si);
printf( ha);
}
```

ANS: hai

Explanation:

- newline
- backspace
- linefeed

Q6. What is the output of the following program?

```
main()
{
41printf(%p,main);
}
```

ANS: Some address will be printed.

Explanation:

Function names are just addresses (just like array names are addresses). main() is also a function. So the address of function main will be printed. %p in printf

specifies that the argument is an address. They are printed as hexadecimal numbers.

Q7. What is the output of the following program?

```
#include
#define a 10
main()
{
#define a 50
printf(%d,a);
}
```

ANS: 50

Explanation:

The preprocessor directives can be redefined anywhere in the program. So the most recently assigned value will be taken.

Q8. Predict the output or error(s) for the following:

```
void main()
{
int const * p=5;
printf(%d,++(*p));
}
```

ANS: Compiler error: Cannot modify a constant value.

Explanation:

p is a pointer to a constant integer. But we tried to change the value of the constant integer.

Q9. What is the output of the following program?

```
main()
{
float me = 1.1;
double you = 1.1;
if(me==you)
printf(I love U);
else
printf(I hate U);
}
```

ANS: I hate U

Explanation:

For floating point numbers (float, double, long double) the values cannot be predicted exactly. Depending on the number of bytes, the precession with of the value represented varies. Float takes 4 bytes and long double takes 10 bytes. So float stores 0.9 with less precision than long double.

Rule of Thumb:

Never compare or at-least be cautious when using floating point numbers with relational operators (== , >, <, <=, >=,!=) .

Q10. What is the output of the following program?

```
#define clrscr() 100
main()
{
clrscr();
printf(%d ,clrscr());
}
```

ANS: 100

Explanation:

Preprocessor executes as a separate pass before the execution of the compiler. So textual replacement of clrscr() to 100 occurs. The input program to compiler looks like this:

```
main()
{
100;
printf(%d ,100);
}
```

Note:
100; is an executable statement but with no action. So it does not give any problem.

Q11. What is the output of the following program?

```
main()
{
char *p=hai friends,*p1;
p1=p;
while(*p!=) ++*p++;
printf(%s %s,p,p1);
}
```

ANS: ibj!gsjfoet

Explanation:

++*p++ will be parse in the given order
_ *p that is value at the location currently pointed by p will be taken
_ ++*p the retrieved value will be incremented
_ when; is encountered the location will be incremented that is p++ will be executed Hence, in the while loop initial value pointed by p is h, which is changed to i by executing ++*p and pointer moves to point, a which is similarly changed to b and so on. Similarly blank space is converted to !. Thus, we obtain value in p

becomes ibj!gsjfoet and since p reaches and p1 points to p thus p1doesnot print anything.

Q12. What is the output of the following program?

```
main()
{
char s[ ]=man;
int i;
for(i=0;s[ i ];i++)
printf( %c%c%c%c, s[ i ], *(s+i), *(i+s), i[s]);
}
```

ANS: mmmm
aaaa
nnnn

Explanation:

s[i], *(i+s), *(s+i), i[s] are all different ways of expressing the same idea. Generally array name is the base address for that array. Here s is the base address. i is the index number/ displacement from the base address. So, indirecting it with * is same as s[i]. i[s] may be surprising. But in the case of C it is same as s[i].

Q13. What is the output of the following program?

```
main()
{
static int var = 5;
printf(%d ,var--);
if(var)
main();
}
```

ANS: 5 4 3 2 1

Explanation:

When static storage class is given, it is initialized once. The change in the value of a static variable is retained even between the function calls. Main is also treated like any other ordinary function, which can be called recursively.

Q14. What is the output of the following program?

```
main()
{
int c[ ]={2.8,3.4,4,6.7,5};
int j,*p=c,*q=c;
for(j=0;j<5;j++) {
printf( %d ,*c);
++q; }
for(j=0;j<5;j++){
printf( %d ,*p);
++p; }
}
```

ANS: 2 2 2 2 2 2 3 4 6 5

Explanation:

Initially pointer c is assigned to both p and q. In the first loop, since only q is incremented and not c, the value 2 will be printed 5 times. In second loop p itself is incremented. So the values 2 3 4 6 5 will be printed.

Q15. What is the output of the following program?

```
main()
{
int i=-1,j=-1,k=0,l=2,m;
m=i++&&j++&&k++||l++;
printf(%d %d %d %d %d, i, j, k, l, m);
}
```

ANS: 0 0 1 3 1

Explanation:

Logical operations always give a result of 1 or 0. And also the logical AND (&&) operator has higher priority over the logical OR (||) operator. So the expression i++ && j++ && k++ is executed first. The result of this expression is 0 (-1 && -1 && 0 = 0). Now the expression is 0 || 2 which evaluates to 1 (because OR operator always gives 1 except for 0 || 0 combination- for which it gives 0). So the value of m is 1. The values of other variables are also incremented by 1.

Q16. What is the output of the following program?

```
main()
{
char *p;
printf(%d %d , sizeof(*p), sizeof(p));
}
```

ANS: 1 2

Explanation:

The sizeof() operator gives the number of bytes taken by its operand. P is a character pointer, which needs one byte for storing its value (a character). Hence sizeof(*p) gives a value of 1. Since it needs two bytes to store the address of the character pointer sizeof(p) gives 2.

Q17. What is the output of the following program?

```
main()
{
int i=3;
switch(i)
{
```

```
default:printf(zero);
case 1: printf(one);
break;
case 2:printf(two);
break;
case 3: printf(three);
break;
}
}
```

ANS: Three

Explanation:

The default case can be placed anywhere inside the loop. It is executed only when all other cases does not match.

Q18. What is the output of the following program?

```
main()
{
char string[]=Hello World;
display(string);
}
void display(char *string)
{
printf(%s,string);
}
```

ANS: Compiler Error: Type mismatch in redeclaration of function display

Explanation:

In third line, when the function display is encountered, the compiler does not know anything about the function display. It assumes the arguments and return types to be integers, (which is the default type). When it sees the actual function display,

the arguments and type contradicts with what it has assumed previously. Hence a compile time error occurs.

Q19. What is the output of the following program?

```
main()
{
int c=- -2;
printf(c=%d,c);
}
```

ANS: c=2;

Explanation:

Here unary minus (or negation) operator is used twice. Same maths rules applies, ie. minus * minus= plus.
Note: However you cannot give like --2. Because -- operator can only be applied to variables as a decrement operator (eg., i--). 2 is a constant and not a variable.

Q20. What is the output of the following program?

```
#define int char
main()
{
int i=65;
printf(sizeof(i)=%d,sizeof(i));
}
```

ANS: sizeof(i)=1

Explanation:

Since the #define replaces the string int by the macro char

Q21. What is the output of the following program?

```
#include
main()
{
char s[]={a,b,c, ,c,};
char *p,*str,*str1;
p=&s[3];
str=p;
str1=s;
printf(%d,++*p + ++*str1-32);
}
```

ANS: 77

Explanation:

p is pointing to character . str1 is pointing to character a ++*p. p is pointing to and that is incremented by one. the ASCII value of is 10, which is then incremented to 11. The value of ++*p is 11. ++*str1, str1 is pointing to a that is incremented by 1 and it becomes b. ASCII value of b is 98.
Now performing (11 + 98 - 32), we get 77(M); So we get the output 77 :: M (Ascii is 77).

Q22. What is the output of the following program?

```
#include
main()
{
struct xx
{
int x=3;
char name[]=hello;
};
struct xx *s;
printf(%d,s->x);
printf(%s,s->name);
```

```
}
```

ANS: Compiler Error

Explanation:

You should not initialize variables in declaration

Q23. What is the output of the following program?

```
#include
main()
{
struct xx
{
int x;
struct yy
{
char s;
struct xx *p;
};
struct yy *q;
};
}
```

ANS: Compiler Error

Explanation:

The structure yy is nested within structure xx. Hence, the elements are of yy are to be accessed through the instance of structure xx, which needs an instance of yy to be known. If the instance is created after defining the structure the compiler will not know about the instance relative to xx. Hence for nested structure yy you have to declare member.

Q24. What is the output of the following program?

```
main()
{
int i=5;
printf(%d%d%d%d%d%d, i++, i--, ++i, --i,i);
}
```

ANS: 45545

Explanation:

The arguments in a function call are pushed into the stack from left to right. The evaluation is by popping out from the stack. And the evaluation is from right to left, hence the result.

Q25. What is the output of the following program?

```
#define square(x) x*x
main()
{
int i;
i = 64/square(4);
printf(%d,i);
}
```

ANS: 64

Explanation:

the macro call square(4) will substituted by 4*4 so the expression becomes i = 64/4*4 . Since / and * has equal priority the expression will be evaluated as (64/4)*4 i.e. 16*4 = 64