C Language Bit Bank

1. **What will be output of the following C program?**

```
#include<stdio.h>
int main(){
        int goto=5;
        printf("%d",goto);
        return 0;
}
```

A) 5
B) 0
C) 1
D) Compilation Error
E) 4

Answer: D

Explanation: Invalid variable name. *goto* is keyword in C language. Variable name cannot be a keyword of C language.

2. **What will be output of the following C program?**

```
#include<stdio.h>
int xyz=10;
int main(){
        int xyz=20;
        printf("%d",xyz);
        return 0;
}
```

A) 10
B) 20
C) 0
D) Compilation Error

Answer: B

Explanation: Two variables can have same name in different scopes. Local Scope is superior to Global scope.

3. **What will be output of the following C program?**

```
#include<stdio.h>
int main(){
    int main = 80;
    printf("%d",main);
    return 0;
}
```

A) Compilation Error
B) 80
C) 0
D) Garbage Value

Answer: B

Explanation: Variable name can be *main*, which is a function and it is not a keyword.

4. **What will be output of the following program?**

```
#include<stdio.h>
int main(){
   int a=5,b = 6;
```

```
    swap( a, b);

    printf("%d %d",a,b);

    return 0;

}

void swap(int a , int b){

int temp = a;

a= b;

b = temp;

}
```

A) 6 5
B) 5 6
C) Garbage Value 5
D) 5 Garbage value
E) No output

Answer: B

Explanation: Scope of a, b is within the function. So, no swap happened to variables.

If you required swapping, you have to swap their addresses.

5. **What will be output of the following program?**

```
#include<stdio.h>
int main(){
   int i=1;
   i=2+2*i++;
   printf("%d",i);
   return 0;
}
```

A) 5
B) 6
C) 8
D) 4

Answer: A

Explanation: i++ i.e. when postfix increment operator is used in any expression, it first assigns its value in the expression then it increments the value of variable by one. So,

i = 2 + 2 * 1

i = 4

Now i will be incremented by one so that i = 4 + 1 = 5.

6. **What will be output of the following program?**

```
#include<stdio.h>
int main(){
   int i=5,j;
   j=++i+++i+++i;
```

```
    printf("%d %d",i,j);
    return 0;
}
```

A) 5 20

B) 5 18

C) Compilation Error

D) 8 24

E) 6 24

Answer: D

Output: **8 24**

Explanation:

Rule : ++ is pre increment operator. So in any arithmetic expression it first increments the value of variable by one in whole expression, then starts assigning the final value of variable in the expression.

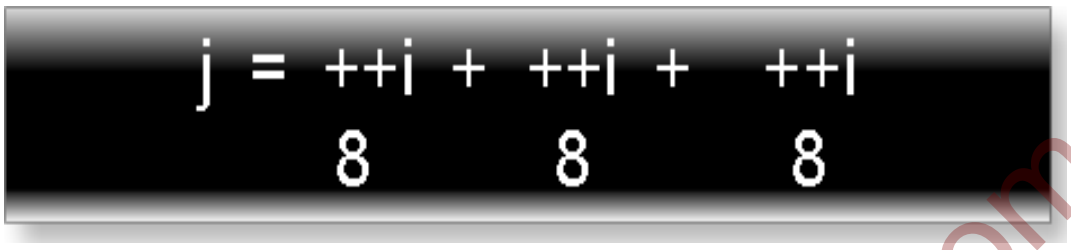
Compiler will treat the expression j = ++i+++i+++i; as

i = ++i + ++i + ++i;

Initial value of i = 5 due to three pre increment operator final value of i=8.

Now final value of i i.e. 8 will be assigned to each variable as shown in

the following figure:



So, j=8+8+8

j=24 and

i=8

7. **What will be output when you execute the following C code?**

```
#include<stdio.h>
void main(){
   int a=5,b=2,c=2;
   if(a&&b<c++){
       printf("TRUE");
   }
   else{
          printf("FALSE");

        }
}
```

A) TRUE
B) FALSE
C) Compilation Error
D) Run time error

Answer: B

8. **What will be output when you execute following C code?**

```
#include<stdio.h>
void main(){
   int a=5,b=10,c=1;
   if(a&&b>c){
      printf("SAKSHI BITBANK");
   }
   else{
      break;
   }
}
```

A) SAKSHI BITBANK
B) Print Nothing
C) Compilation Error

Answer: C
Explanation: Keyword break is not syntactical part of if-else statement. So we cannot use break keyword in if-else statement. This keyword can be used in case of loop or switch case statement.

Hence when you will compile, above code compiler will show an error message: *Misplaced break*.

9. **What will be output when you execute following C code?**

```
#include<stdio.h>
void main(){
   int a=5,b=10;
   if(++a||++b)
      printf("%d  %d",a,b);
   else
      printf("HTST Quiz");
}
```

A) 5 10
B) 6 10
C) 6 11
D) 5 11
E) HTST Quiz

Answer: B

Explanation: Consider the following expression:

++a || ++b

In this expression || is Logical OR operator. Two important properties of this operator are:

Property 1:

(Expression1) || (Expression2)

|| operator returns 0 if and only if both expressions return a zero otherwise it || operator returns 1.

Property 2:

To optimize the execution time there is rule, Expression2 will only evaluate if and only if Expression1 return zero.

In this program initial value of a is 5. So ++a will be 6. Since ++a is returning a non-zero so ++b will not execute and if condition will be true and if clause will be executed.

10. **What is output of the following program?**

```
#include<stdio.h>
void main(){
   int a=5,b=10;
   if(++a && ++b)
       printf("%d  %d",a,b);
   else
       printf("Sakshi");
}
```

A) 5 10
B) 6 10
C) 6 11
D) 5 11
E) Sakshi

Answer: C

Explanation: Consider the following expression:

++a && ++b

In this expression && is Logical AND operator. Two important properties of this operator are:

Property 1:

(Expression1) && (Expression2)

&& operator returns 0 either of the expressions return a zero.

Property 2:

Here Both expressions should evaluate not like previous, based on evaluation the result will display. So, both will evaluate.

11. **What will be the output when the following C code is executed?**

```
#include<stdio.h>
void main(){
   int x=-1,y=-1;
   if(++x=++y)
       printf("DennisRitche");
   else
       printf("JamesGosling");
}
```

A) DennisRitche
B) JamesGosling
C) Warning: Condition is always true
D) Compilation error

Answer: D

Explanation: Consider following statement:

++x=++y

As we know ++ is pre increment operator in the above statement. This operator increments the value of any integral variable by one and returns that value. After performing pre-increments, above statement will be:

0=0

In C language, it is illegal to assign a constant value to another constant. Left side of = operator must be a container i.e. a variable. So compiler will show an error message: *L value required.*

If we want to make it true…we have to use == (Equal to operator) = is assignment operator.

12. **What will be output when you will execute following C code?**

```
#include<stdio.h>
void main(){
   if(0xA)
      if(052)
         if('\xeb')
            if('\012')
               printf("SAKSHI EDUCATION");
            else;
         else;
      else;
   else;
}
```

A) SAKSHI EDUCATION
B) Print Nothing
C) Compilation error: Misplaced else
D) Compilation error: If without any body

E) Compilation error: Undefined symbol

Answer: A

Explanation:

oxA: It is hexadecimal integer constant.

052: It octal integer constant.

'\xeb': It is hexadecimal character constant.

'\012': It is octal character constant.

As we know in C language, zero represents false and any non-zero number represents true. All of the above constants return a non-zero value. So all if conditions in the above program are true.

In C, it is possible to write else clause without any body.

13. **What will be output when you will execute following C code?**

```
#include<stdio.h>
void main(){
   int x=1;
   if(x--)
      printf("X = %d"+x);
      --x;
   else
      printf("%d",x);
}
```

A) 0
B) 1

C) Compilation Error
D) -1

Answer: C

Explanation: If you are not using **{**and**}** in if clause then you can write only one statement. Otherwise it will cause of compilation error: *Misplace else*

14. **What will be output when you execute following C code?**

```
#include<stdio.h>
void main(){
   int a=2;
   if(a--,--a,a)
      printf("I am Good");
   else
      printf("You are Good");
}
```

A) I am Good
B) You are Good
C) Compilation error: Multiple parameters in if statement
D) Run Time Error

Answer: B

Explanation: Consider the following expression:

a-- , --a , a

In C language, comma behaves as separator as well as operator. In the above expression comma is behaving as operator. Comma operator enjoys least precedence in precedence table and its associativity is left to right. So first of all left most comma operator will perform operation

	then right most comma will operator in the above expression. After performing a-- : a will be 2 After performing --a : a will be 0 a=0 As we know in C language, zero represents false and any non-zero number represents true. Hence else part will execute.

15.	**What will be output when you execute following C code?**
	`#include<stdio.h>`
	`void main(){`
	`int x=1,y=-1,z =0,;`
	`if(x==y)`
	`printf("Equal");`
	`else`
	`printf("Not Equal");`
	`if(z)`
	`printf("True");`
	`else`
	`printf("False");`
	`}`
	A) Equal True
	B) Not Equal True
	C) Not Equal False
	D) Equal False

Answer: C

16. **What will be output when you will execute following C code?**

```
#include<stdio.h>
void main(){
    int check=2;
    switch(check){
      case 1: printf("Sachin");
      case 2: printf("Dravid");
      case 3: printf("Lakshman");
      default: printf("Dhoni");
    }
}
```

A) Dravid
B) Dhoni
C) Dravid, Lakshman, Dhoni
D) Compilation Error
E) Sachin, Dravid, Lakshman, Dhoni

Answer: C

Explanation: Execute from second case statement. As we don't have break statement all other statements will execute.

17. **The unused memory is released by using**

A) release
B) free
C) malloc
D) None of the above

Ans: B

18. **What will be output when you will execute following C code?**

```
#include<stdio.h>
void main(){
        int x = 0;
        for (x=1; x<4; x++);
        printf("x=%d\n", x);
}
```

A) X= 0,X= 1,X= 2,X= 3, X=4
B) X= 1,X= 2,X= 3, X=4, X=5
C) X=0
D) X=4
E) X =5

Answer: E

Explanation: for loop have; at it's end. So printf function is not under for loop. So prints 5.

19. **int x; /* assume x is 16 bits in size */**

What is the maximum number that can be printed using printf("%d\n", x), assuming that x is initialized as shown above?

A) 127
B) 128
C) 32768
D) 32767
E) 65536

	Answer: D **Explanation:** Range of int = 2 Byte 2 Bytes = 2*8 Bits ➔ 16 Bits 1 bit for sign, so 16-1 = 15. -2^15 to 2^15 -1 (-1 because including 0)
20.	**What will be output when you will execute following C code?** **#include<stdio.h>** **void main()** **{** **int s=0;** **while(++s<10)** **{** **if(s<4 && s<9)** **continue;** **printf("\n%d\t",s);** **}** **}** A) 1 2 3 4 5 6 7 8 9 B) 1 2 3 10 C) 4 5 6 7 8 9 10 D) 4 5 6 7 8 9 E) 5 6 7 8 9 **Answer: D**

21. **What will be output when you will execute following C code?**

```
#include<stdio.h>
void main()
{
  int s=0;
  while(s++<10)
  {
    if(s<4 && s<9)
      continue;
    printf("\n%d\t",s);
  }
}
```

A) 1 2 3 4 5 6 7 8 9
B) 1 2 3 10
C) 4 5 6 7 8 9 10
D) 4 5 6 7 8 9
E) 5 6 7 8 9

Answer: C

Explanation:

At While loop Starting the value of S is 0 only…in next step it is 1. Like that it if s = 9 then while(S++<10) will execute first and in block of the loop the value become 10. So print Even 10.

22. **What will be output when you execute following C code?**

```
#include<stdio.h>
void main()
{
  int s=0;
  while(++s<10)
  {
    if(s<4 && s<9)
      continue;
    printf("\n%d\t",s);
  }
}
```

A) 1 2 3 4 5 6 7 8
B) 1 2 3 10
C) 4 5 6 7 8 9 10
D) 4 5 6 7 8 9
E) 5 6 7 8 9

Answer: D

23. **What will be output when you execute following C code?**

```
#include<stdio.h>
void main()
```

```
{
  int s=0;
  while(s<10)
  { (s++<4)
     continue;
    printf("\n%d\t",s);
  }
}
```

A) 1 2 3 4 5 6 7 8 9
B) 1 2 3 10
C) 4 5 6 7 8 9 10
D) 5 6 7 8 9 10
E) 5 6 7 8 9

Answer: D

24. **What will be output when you will execute following C code?**

```
void main()
{
  int a=10,b=20;
  char x=1,y=0;
  if(a,b,x,y)
  {
    printf("EXAM");
  }
}
```

A) EXAM
B) 0
C) Nothing will be printed.
D) Compilation Error

Answer: C

25. **What will be output if you will compile and execute the following C code?**

```
#define x 5+2

void main(){

   int i;

   i=x*x*x;

   printf("%d",i);

}
```

A) 125
B) 27
C) 8
D) 343

Answer: B

Explanation: As we know #define is token pasting preprocessor it only paste the value of micro constant in the program before the actual compilation start.

You can absorb #define only pastes the 5+2 in place of x in program. So,

i=5+2*5+2*5+2

=5+10+10+2

=27

26. **What will be output if you compile and execute the following C code?**

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

A) Equal
B) Greater than
C) Less than
D) Compiler Error

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.

27. **What will be output if you execute following C code?**

```
#include<stdio.h>
int main(){
   int i;
```

```
    for(i=0;i<5;i++){
        int i=10;
        printf(" %d",i);
        i++;
    }
    return 0;
}
```

A) Compilation Error
B) 10 11 12 13 14
C) 10 10 10 10 10
D) 0 1 2 3 4 5

Answer: C

Explanation: Local variable is more precious than global.

28. **What will be output of following program?**

```
#include<stdio.h>
void main(){
        int i = 5 , j;
        int *p , *q;
        p = &i;
        q = &j;
        j = 5;
        printf("value of i : %d value of j : %d",*p,*q);
        getch();
}
```

A) 5 5
B) Address Address
C) 5 Address
D) Compilation Error
E) Run Time Error

Answer: A

Explanation: P is pointer variable stores the address location of an integer variable. Like that q as well.

29. **What is the output of the following program?**

```
#include<stdio.h>
void main(){
        int *p1;
        long double *p2;
        printf("%d   %d",sizeof(p1),sizeof(p2));
}
```

A) 2 4

B) 2 8

C) 2 2

D)4 4

Answer: C

Explanation: Size of any type of pointer is independent of the data type which is it is pointing i.e. size of pointer is always fixed. Size of any type

(near) of pointer in C language is two byte.

Since both pointers *int* and *long double* are pointing to only first byte of *int* data and *long double* data respectively.

Hence both *int* pointer and *long double* pointer stores only address in 16 bits. Thus both of them will occupy exactly equal memory space.

30. **What will be output of following program?**

```
#include<stdio.h>
void main(){
        int  a = 10;
        void *p = &a;
        int *ptr = p;
        printf("%u",*ptr);
        getch();
}
```

A) Compilation Error
B) 10
C) Address
D) Run Time Error
E) 2

Answer: B

Explanation: Void pointer can hold address of any data type without type casting. Any pointer can hold void pointer without type casting.

31. **What will be output of following program?**

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

```
void main(){
  int i = 3;
  int *j;
  int **k;
  j=&i;
  k=&j;
  printf("%u %u %d ",k,*k,**k);
}
```

A) 2 2 2

B) Address Address 3

C) Address 3 3

D) 3 3 3

Answer: B

Explanation: Memory representation

Here 6024, 8085, 9091 is any arbitrary address, it may be different.

Value of k is content of k in memory which is 8085.

	Value of *k means content of memory location which address k keeps. k keeps address 8085 . Content of at memory location 8085 is 6024 In the same way **k will equal to 3. Short cut way to calculate: Rule: * and & always cancel to each other i.e. *&a = a So *k = *(&j) since k = &j *&j = j = 6024 And **k = **(&j) = *(*&j) = *j = *(&i) = *&i = i = 3

32.	**What will be output of following program?**
	`#include<stdio.h>`
	`#include<string.h>`
	`void main(){`
	`register a = 25;`
	`int *p;`
	`p=&a;`
	`printf("%d ",*p);`
	`getch();`
	`}`
	A) 25
	B) 4
	C) Address

D) Compilation Error

Answer: D

Explanation: Register data type stores in CPU. So it has not any memory address. Hence we cannot write &a.

33. **What will be output of following program?**

```
#include<stdio.h>
void main(){
        int i = 5;
        int *p;
        p = &i;
        printf(" %u %u", *&p , &*p);
        getch();
}
```

A) 5 Address

(B) Address Address

(C) Address 5

(D) Compilation error

(E) None of above

Answer: B

Explanation:

Since * and & always cancel to each other.

	i.e. *&a = a so *&p = p which store address of integer i &*p = &*(&i) //since p = &i = &(*&i) = &i So second output is also address of i.

34.	**What will be output of following program?**
	`#include<stdio.h>`
	`void main(){`
	`int i = 100;`
	`printf("value of i : %d addresss of i : %u",i,&i);`
	`i++;`
	`printf("\nvalue of i : %d addresss of i : %u",i,&i);`
	`getch();`
	`}`
	(A)
	value of i : 100 addresss of i : Address
	value of i : 101 addresss of i : Address
	(B)
	value of i : 100 addresss of i : Address
	value of i : 100 addresss of i : Address
	(C) value of i : 101 addresss of i : Address
	value of i : 101 addresss of i : Address

(D) Compilation error

(E) None of above

Answer: A

Explanation:

Within the scope of any variable, value of variable may change but its address will never change in any modification of variable.

35. **What will be output of following program?**

```
#include<stdio.h>
void main(){
        int i = 3;
        int *j;
        int **k;
        j = &i;
        k = &j;
        printf("%u %u %u",i,j,k);
}
```

(A) 3 Address 3

(B) 3 Address Address

(C) 3 3 3

(D) Compilation error

(E) None of above

Answer: B

Explanation:

ere 6024, 8085, 9091 is any arbitrary address, it may be different.

36. **What will be output of following program?**

```
void main(){
        int *ptr=( int *)1000;
        ptr=ptr+1;
        printf(" %u",ptr);

}
```

A) 1000

B) 1001

C) 1002

D) Compilatation Error

Answer: C

Explanation:

Address + Number= Address

Address - Number= Address

Address++ = Address

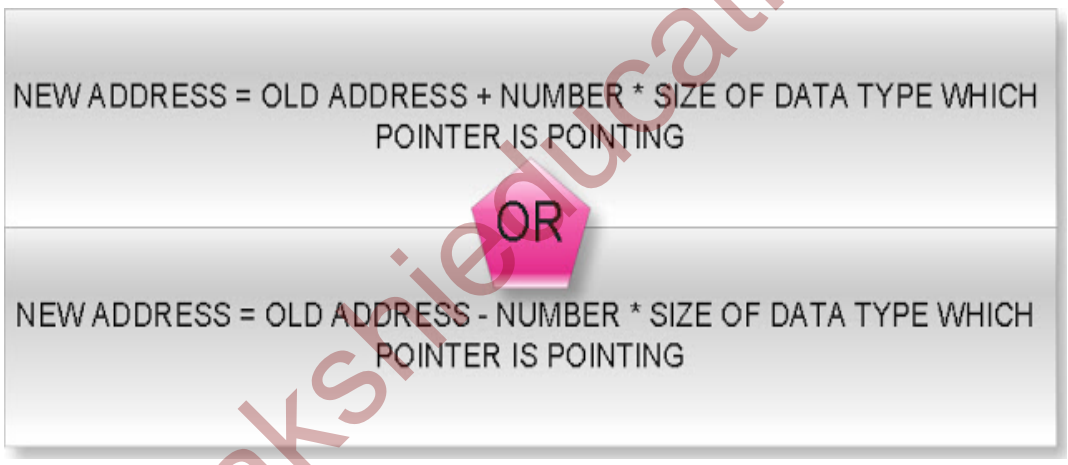
Address-- = Address

++Address = Address

--Address = Address

If we will add or subtract a number from an address result will also be an address.

New address will be:



37. **What will be output of following C program?**

```
void main(){
        double *p=(double *)1000;
        p=p+3;
        printf(" %u",p);
}
```

A) 1000

B) 1002
C) 1024
D) 1003
E) Compilation Error

Answer: C

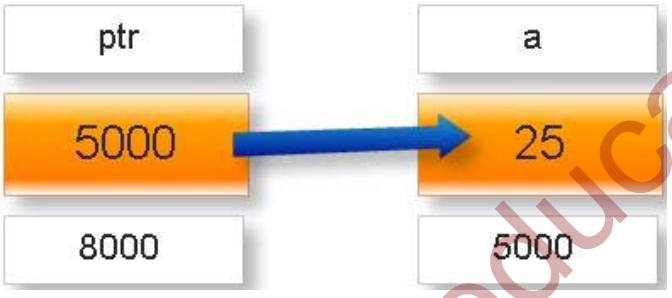
Explanation: 1000+3*8 Bytes

38. **What will be output of following C program?**

```
int *call();

void main(){

        int *ptr;

        ptr=call();

        clrscr();

        printf("%d",*ptr);

}

int * call(){

        int x=25;

        ++x;

return &x;

}
```

Output: Garbage value.

Explanation: Variable x is local variable. Its scope and lifetime is within the function call hence after returning address of x variable x became dead and pointer is still pointing ptr is still pointing to that location.

	Solution of this problem: Make the variable x is as static variable. In other word we can say a pointer whose pointing object has been deleted is called dangling pointer.
39.	**What is Dangling pointer?** If any pointer is pointing the memory address of any variable but after some variable has deleted from that memory location while pointer is still pointing such memory location. Such pointer is known as dangling pointer and this problem is known as dangling pointer problem. **Initially:**  **Later:**