1. Find the output for the following programs(branching and looping)

#include<stdio.h>

Void main()

{

int i;

for( i = 1 ; i < 4 ; i++)

{

switch(i)

{

case 1 : printf("%d" , i);break;

case 2 : printf("%d" , i);break;

case 3 : printf("%d" , i);break;

}

}

switch(i)

{

case 4 : printf("%d" , i);break;

}

}

Output : 1234

explanation : loop 3 time execute aagum athula i vanthu 1 la irunthtu 3 varaikkum swich la pogum athula i la 123 nu print aagum .but loop break aagum pothu i vanthu 4 aagi irukum so next switch case success aagi 4 um print aagum.

1. Find the output( operartor and expression)

void main()

{

char \*s = "\12345s\n";

printf("%d" , sizeof(s));

}

Output : 4

explanation: ithula s la irukra value pakka theva illa because namba print panrathu

sizeof(s) so pointers oda size 4 byte so 4 nu print aagum.

1. Find the output( Funtions)

int main()

{

static int i = 3;

printf("%d" , i--);

return i>0 ? main() : 0 ;

}

Output : 321

explanation : initial la i oda value 3 .first time 3 value 3 print aagum aprm i-- aagi i=2 aagum ithula 2>0 condition true so again main function ah call pannum ippo 2 print aagum aprm i-- aagi i=1 aagum 1>0 condition true so again main function ah call pannum ippo 1 print aagum i= 0 aagidum ippo 0>0 conditon false so program exit aagidum.

1. Find the output(pointers)

int main()

{

char \*s[]={ "dharmr'a","hewlett-packard","siemens","ibm"};

char \*\*p;

p = s ;

printf("%s" ,++\*p);

printf("%s",\*p++); ;

printf("%s" ,++\*p);

}

Output: harmr’aharmr’aewlett-packard

exaplanation: input vanthu array of character pointers so oru oru word um array va irukum so p=s apdi assign pannum pothu “dharmr’a” oda address vanthu p point pannum ithula \*p print panna “dharmr’a” print aagum aana namba pre -increment panrathunala p “dharmr’a” la irukra h ah point pannum so print panna “harmr’a” nu print aagum aprm next line post-increment so first p point pannitu irukra “harmr’a” print aagitu array increment aagidum so ippo p “hewlett-packard” point pannum ippo p pre-increment aagurathunala p inrement aagi e character point pannum ippo print pannum pothu “ewlett-packard” print aagum.

1. Find the output( dynamic memory)

#include<stdio.h>

#include<malloc.h>

#include<string.h>

int main()

{

int i;

char a[]="String";

char \*p = "New String";

char \*temp;

temp = malloc(strlen(p) + 1);

p = malloc( strlen(temp) + 1);

strcpy(p , temp);

printf("%s" , p);

}

Output : unpredictable string

explanation : malloc na memory allocation so temp ku memory allocate panrom evlo na p length 10 and 10+1 =11 so 11 byte create aagum aprm p kum temp oda length 0 so 0+1=1 p= 1 byte create aagum strcpy na string copy so p 1 byte la temp 11 byte ah copy panrom temp vanthu empty ah irukrathunaala garbage value varum athanala unpredictable string.

1. Find the output(algorithm)

int main()

{

int n = 12 , res = 1;

while( n > 3)

{

n -= 3;

res \*= 3;

}

printf("%d" , n\*res);

}

Output : 81

explanation:initial n=12 12>3 so n=12-3; n=9 aagidum res=1\*3=3 aagidum again 9>3 condition true n=9-3=6 aagum res=3\*3=9 aagum again 6>3 condition true n=6-3=3; res=9\*3=27 again 3>3 condition false exit aagum print pannum pothu n and res multiple pannum pothu 3\*27 aagum so ans is 81.

1. Find the output(function)

void fun(int [][3]);

int main()

{

int a[3][3] = {9,8,7,6,5,4,3,2,1};

fun(a);

printf("%d\n" , a[2][1]);

}

void fun(int b[][3])

{

++b;

b[1][1]=5;

}

Output : 5

expalanation:iniially 2 D array {9,8,7

6,5,4

3,2,1 };

ithu 2D array a first 1 D array ahavahu {9.8.7} oda address ah point pannitu irukum itha pass panni pre-increment panna (++b) next {6,5,4} oda address ah point pannum so ippo array epdi irukum na b[2][3] so ithula b[1][1] na 2 va 5 ah mathum athvathu a array la a[2][1] oda address so main function la a[2][1] ah print panna ah nu print aagum.

1. Find the output(strings)

void main()

{

int i , n;

char x[5];

strcpy( x , "Zoho");

n = strlen(x);

\*x = \*(x+(n-1));

printf("%s" , x);

}

Output: ooho

explanation: strcpy pannum pothu x[0]=z , x[1]=0,x[2]=h,x[3]=o irukum.

Next n =strlen(x) pannum pothu x=4 length of string x next line x base address ah point pannum so n-1 na 3 .base address + 3 so x oda last o va base address athavathu z irukra address ku update pannidum so x ah print panna ah ooho nu print aagum.

1. Find the output(arrays)

void main()

{

int c[]={5,4,3,4,5};

int j , \*q = c;

for( j = 0 ; j<5 ; j++){

printf("%d" , \*c);

++q;

}

}

Output:55555

explanation: character array c atha oru q apdinra pointer ku copy panrom c vanthu array oda base address athavathu 5 va point pannitu irukum so print panna 5 time um 55555 tha print aagum because namba copy panna q variable la tha increment panrom athu c ah affict pannathu.

1. Find the output(branching and looping)

void main()

{

int i = 1;

for(i =0 ; i= -1 ; i=1){

printf("%d", i);

if(i!= 1) break;

}

}

Output: -1

explanation: initial i=1 after loop initialization now i=0 after condition checking i=-1 so print pannum pothu -1 print aagum if la check pannum pothu -1!=1 condition true so break aagi exit aagidum.

1. Find the output(Arrays)

void main()

{

int s[] = {1,0,5,0,10,0};

int f[] = {2,4,6,8,10,12};

int n = 6 , i = 0 , j = 0;

for( j = 1 ; j < n ; j++)

{

if( s[j] >= f[i])

{

printf("%d" , i);

i = j;

}

}

}  
output : 02

explanation:first time 0>2 condition false ethuyum aagathu next 5>2 true 0 print aagum i=2 aagidum next 0>6 false next 10>6 true 2 print aagum i=4 now 0>10 false next length condition false loop terminate aagidum so 02 nu print aagum.

1. Find the output(Functions)

void f(int \*a , int m)

{

int j = 0;

for(j = 0 ; j < m ; j++)

{

\*(a+j) = \*(a+j) - 5;

}

}

void main()

{

int a[] = {'f' , 'g' , 'h' , 'i' , 'j'};

int j = 0 ;

f(a , 5);

for(j = 0 ; j<= 4 ; j++)

printf("%c\t" , a[j]);

}

Output:a b c d e

explanation:a array and 5 va function first time a array la first character f ah edukum f-5 pannum pothu ascii value f ku 102-5=97 athavathu a atha a[0]=a store panrom next g-5 = b atha a[1] = b nu store panrom nect h-5=c athu a[2]=c nu store aagum ithe mathiri innum 2 character num pannanum next print panna a b c d e apdinu print aagum.

1. Find the output(branching and looping)

void main()

{

int i=0, j=0 , sum=0;

for(i= 1; i < 500 ; i\*=3)

for(j=0;j<i;j++)

sum++;

printf("%d",sum);

}

Output: 364

explanation: initial sum=0 first time j=0 i=1 so second loop oda condition padi true sum=1 aagum next j=1 aagi condition false aagum ithu mathiri i time j loop execute aagum ithula mukkiyama i=i\*3 aaguthu atha note pannaum so second time i=3 and 9,27,81,…

1. Find the output(branching and looping)

void main()

{

int n;

for(n = 6 ; n!= 1; n--)

printf("%d" , n--);

}

Output: infinite loop

n vanthu decrement 2 times nadakkuthu so athu even aavetha irukume thavira epppathu 1 aaga chance illa so infinite loop.

1. Find the output(arrays)

void main()

{

int a[3][4] = {2,4,6,5,10,12,12,10,5,6,4,2};

int i = 0 , j , k =99;

while(i < 3)

{

for(j = 0 ; j < 4 ; j= j++)

{

if( a[i][j] < k)

{

k = a[i][j];

}

}

i++;

}

printf("%d" , k);

}

Output : 2

explanation: a={2,4,6,5

10,12,12,10

5,6,4,2

};

outer while loop 3 time and inner for loop 4 time execute aagum and k=99

first 2<99 true so k=2 nu aagidum next 4<2 false 6<2 and 5 <2 false so ippo i++ aagum so array la patha 2 tha minimum athanala inimel k chance aaga vaippu illa so 2 nu print aagum.

1. Find the output( pointer)

void main()

{

char \*x="Alice";

int i , n = strlen(x);

\*x = x[n];

for(i=0; i<=n; i++)

{

printf("%s ", x); x++;

printf("\n", x);

}

return 0;

}

Output : runtime error

n=5 so x[5] vathu null irukum because 0 to 4 varikkum tha x la values irukum 5 null ah irukum so null x ah increment panrathu naala runtime error.

1. Find the output(structures and union)

struct value{

int bit1:1;

int bit3:4;

int bit4:4;

}bit;

int main()

{

printf("%d\n", sizeof(bit));

return 0;

}

Output : 4 bytes

explanation: ithula struct ku 9 bits than thevai but minimum ah 4bytes that is 32 bits allocate pannum. Size of operator always return bytes.

1. Find the output(dynamic memory)

struct node // intha struct oda size is 16 bytes

{

int data; // 4 bytes

float d; // 4 bytes

struct node \*link; // 8 bytes

};

int main()

{

struct node \*p, \*q;

p = (struct node \*) malloc(sizeof(struct node));

q = (struct node \*) malloc(sizeof(struct node));

printf("%d, %d\n", sizeof(p), sizeof(q));

return 0;

}

Output : 4 , 4

Explanation: inga sizeof operator pointer oda size ah mattum the return pannum not returns the sizeof structure it points to. 32-bit system la pointer oda size is 4 so 4,4 nu output varuthu 64-bit system la pointer oda size 8 so antha systems la lam 8,8 nu output varum.

1. Find the output(structures and unions)

typedef union

{

int a;

char b[10];

float c;

}Union;

int main()

{

Union x , y = {100};

x.a = 50;

strcpy(x.b , "Hello");

x.c = 21.50;

printf("%d %s %f\n" , x.a , x.b , x.c);

printf("%d %s %f" , y.a,y.b, y.c);

}

Output:1101791232 21.500000

100 d 0.000000

1. Find the output(structures and union)

struct point{

int x;

int y ;

};

struct point origin , \*pp;

int main()

{

pp = &origin;

printf("origin is (%d %d)\n", (\*pp).x , (\*pp).y);

printf("origin is (%d %d)" , pp->x , pp->y);

return 0;

}

Output : origin is (0 0 )

origin is (0 0 )

1. Find the output(branching and looping)

void main()

{

int i = -1;

printf("i =%d +i = %d\n" , i , +1);

}

Output : i=-1 i=1

1. Find the output(datatypes)

void main()

{

char not;

not=12;

printf("%d",not);

}

Output : 12

1. Find the output(branching and looping)

#define FALSE -1

#define TRUE 1

#define NULL 0

void main()

{

if(NULL)

puts("NULL");

else if(FALSE)

puts("TRUE");

else

puts(" FALSE");

}

Output : TRUE

1. Find the output(operator and expressions)

void main()

{

int k = 1;

printf("%d==1 is"" %s" ,k, k == 1 ? "TRUE":"FALSE");

}

Output : 1==1 is TRUE

1. Find the output(file manipulation)

int main()

{

FILE \*ptr;

char i;

ptr=fopen("demo.c","r");

while((i=fgetch(ptr))!=EOF)

printf("%c",i);

}

1. Find the output(branching and looping)

int main()

{

int t , i ;

for ( t=4;scanf("%d",&i)-t;printf("%d\n",i))

printf("%d--",t--);

}

Output : loop runs 4 times

5 // 5 is the user input here

4--5

5

3--5

5

2--5

5

1. Find the output(structures and unions)

struct emp{

int len;

char name[1];

};

int main()

{

char newname[] = "Rahul";

struct emp \*p = (struct emp \*) malloc(sizeof(struct emp) -1 + strlen(newname)+ 1);

p->len = strlen(newname);

strcpy(p -> name, newname);

printf("%d %s\n", p->len, p->name); return 0;

}

Output : 5 Rahul

Explanation : p.len = 5 and p.name = “R” mattum the store agum, since %s iruka nala “R” oda address la irunthu antha string full ah print agum

1. Find the output(algorithm)

int main() {

printf("%d %d %d %d\n",72,072,0x72,0X72);

return 0;

}

Output : 72 58 114 114

1. Find the output(operator and expression)

void main()

{

char ch;

int a;

float b;

printf("bytes occupied by ch=%d\n",sizeof(ch));

printf("bytes occupied by a=%d\n",sizeof(a));

printf("bytes occupied by b=%d\n",sizeof(b));

}

Output :

Bytes occupied by ch=1

Bytes occupied by a=4

Bytes occupied by b=4

1. Find the output(operator and expressions)

void main()

{

printf("%d\n" , sizeof('7')); // op: 4

printf("%d\n" , sizeof('a')); // op: 4

printf("%d\n",sizeof("a")); // op: 2

printf("%d\n", sizeof("ab")); // op: 3

printf("%d\n", sizeof("abcd")); // op: 5

printf("%d\n" , sizeof(7)); // op: 4

printf("%d\n" , sizeof(7.0f)); // op: 4

printf("%d\n" , sizeof(7.0)); // op: 8

}Output: 4 here it takes sizeof(‘7’) as sizeof(int) so it gives 4 as output

4 sizeof(int) is 4 so the op is 4

8 here it takes 7.0 as double so the sizeof(double) is 8.

1. Find the output(datatypes)

void main()

{

char ch=291;

printf("%d %d %c\n",2147483648,ch,ch);

return 0;

}

Output : -2147483648 35 #

Explanation: the range of int is -2147483648 to 2147483647 so -2147483648 is the 1st output. 35 is produced because the range of char is from -128 to 127 for unsigned and for signed is 256 so 291%256=35. # is produced because the ascii value of 35 is #.

1. Find the output(datatypes)

void main()

{

int g;

g=300000\*300000/300000;

printf("g=%d\n",g);

}

Output : -647

1. Find the output(datatypes)

void main()

{

float a;

a=4/2;

printf("%f %f\n",a,4/2);

}

Output : 2.000000 0.000000

1. Find the output(operator and expression)

void main()

{

printf("%d\n",sizeof(4)/sizeof(2.0));

printf("%d\n",sizeof(2.0)/sizeof(4));

}

Output : 0 2

Explanation: 0=>(4/8) and 2=>(8/4)

1. Find the output(operator and expression)

void main()

{

int x=10,y=5,p,q;

p=x > 9;

q=x>3&& y!=3;

printf("p=%d q=%d \n",p,q);

}

Output : p = 1 q=1

36.

#include<stdio.h>

void main(){

float x=0.1;

if(x==0.1){

printf("If");

}

else if(x==0.1f){

printf("else if");

}

else{

printf("else");

}

}

Output: else if

37.

int main(){

int a[10]={9,8,2,3,1};

int i,j,m;

i=a[0];

j=a[i];

m=a[j];

printf("%d,%d,%d",i,j,m);

return 0;

}

Output: 9,0,9

38.

#include<stdio.h>

#include<string.h>

int main(){

char str[9]={'z','o','h','o','c','o','r','p'};

int i,j,len;

len=strlen(str);

printf("len is:%d\n",len);

for(i=0,j=1 ; (i+j)<len; i++)

{

if(str[i] < str[i+j]){

str[i] = str[i]+1;

}

else{

str[i+j] = str[i+j] + 1;

}

}

printf("%s",str);

return 0;

}

Output:

Len is: 8

zpjoeprq

39.

#include<stdio.h>

int main(){

int i=2,j=2;

while(i+1?--i:j++){

printf("%d",i);

}

return 0;

}

Output: 1

Explanation:

The while expression, i+1?--i:j++, first checks to see if i+1 is non-zero. If so, then it evaluates --i, using it as the while test. If not, it evaluates j++, using it as the while test.

Initially i is 2, so i+1 is 3 which is non-zero. So it evaluates --i, which is 1, and i is now 1. It then executes the body, printing 1.

The second time around, i is 1, so i+1is 2 which is non-zero. So it again evaluates --i, which is 0, and i is now 0. It then exits the loop, since the value of the while expression is 0.

40. The output should be lighthouse complete the if condition?

#include<stdio.h>

void main()

{

if(\_\_\_?????\_\_\_){

printf("light");

}

else{

printf("house");

}

}

Output: if( printf("light")?0:1 )

41.

#include<stdio.h>

int main(){

int i=5;

printf("%d,%d,%d,%d,%d\n",i++,i--,++i,--i,i);

return 0;

}

Output:

EXPLANATION-1 :

The logic/rules made should take us to the answer 45545 but compiler does also give answer as 45555

45545  
  
because the expiration will executed from right to left and will print from left to right.  
  
at the execution time from right i=5,--i=4,++i=5,i--=5 then decremented to4,now i++=4 ,then incremented to 5.

left to right 45545

EXPLANATION-2:

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.  
  
So the output will be 45545.

EXPLANATION -3:

For every pre-increment and pre-decrement and 'i' values are replaced with final 'i' value.i.e,5 here.

First thing we will operate statement from right to left to get assigned value, i. e from i to i++  
After operation we will put the assigned value from left to right  
  
  
  
Second thing is in pre decrement/pre increment/variable doesn't have any pre/post fix, i.e in case off --i, ++i & i the value will assign in the end of the statement.  
  
  
As we move towards right to left  
First we take i, which has value 5  
Assign value is expected to be 5 but it will remains empty because of 2nd rule, we will assign value in end  
Our update value is still 5  
  
Second we take --i, it is a pre decrement, means value will decrement first, then it will assign  
So from our updated value, we have i=5  
Decrement value is 5-1=4  
Assign value is expected to be 4 but it remains empty because for second rule  
Our updated value is now 4  
  
Third we take ++i, it is pre increment means value will increment first, then it will assign.  
So from our updated value, i=4  
Increment value is 4+1=5  
Assign value is expected to be 5 but it remains empty because of 2nd rule  
Our update value is now 5  
  
  
Fourth we take i--, it is post decrement means value will assign first, then it will decrement  
From our updated value, we have i=5  
Assign value is 5  
Then decrement value is 5-1=4  
So our new updated value is now 4  
  
  
Lastly we have I++, post increment means value will assign first, then it will increment.  
Our updated value is 4  
Assign value is 4  
Increment value is 4+1=5  
Our new updated value is 5 and this is our final newest updated value  
  
Now look for all 5 assign value  
First assign value is empty  
Second assign value is empty  
Third assign value is empty  
Fourth assign value is 5  
Fifth assign value is 4  
  
Now put our newest update value I, e 5 to empty assign value  
  
So our value will  
1st assign value 5  
Second is 5  
Third is 5  
Fourth is 5  
Fifth 4  
  
And put assign value from left to right you get 45555.

42.

#include<stdio.h>

int main(){

int a=10;

printf("%d, %d, %d\n",++a, a++,a);

return 0;

}

Output: 12, 10, 12

Explanation : Operation is executed from right to left (stack).

Step 1: a = 10

Step 2: a++ = 10, then 10 is incremented to 11

Step 3: ++a = 12

Final value of a is 12.

After execution result is printed from left to right. Here, keep the values calculated for post increment and post decrement same as it is (obtained from the above steps). Whereas, update the value of ‘a’ and ‘- -a’ and ‘++a’ to the final value of a (i.e., here final value of a is 12).

So, the final output is 12(final value of a), 10(since it is a post increment/ decrement), 12(final value of a).

43.

#include<stdio.h>

int main(){

int a,b;

for(a=6,b=4; a<=24; a=a+6){

if(a%b==0){

break;

}

}

printf("%d",a);

return 0;

}

Output: 12

44.

#include<stdio.h>

int main(){

int val=2;

do{

val++;

++val;

}while(val++ ==4 );

printf("%d\n",val);

return 0;

}

Output: 8

45.

#include<stdio.h>

void fun(int n, int m){

if(n>0){

fun(--n,++m);

printf("%d,%d\n",n,m);

fun(--n,++m);

}

}

int main(){

int a=3;

fun(a,a);

return 0;

}

Output:

0,6

1,5

2,4

0,6

46.

#include<stdio.h>

int fun(int val){

int i;

int ans = val;

for(i=0; i<val; i++){

ans = ans + (val\*val);

}

return ans;

}

int main(){

int i[5] = {0,1,2,3,4};

int n=0;

do{

printf("%d,\t",fun(i[n]));

}while(++n < 5);

return 0;

}

Output: 0, 2, 10, 30, 68

47.

#include<stdio.h>

int main(){

static int staticVar;

int j;

for(j=0; j<=5; j+=2){

switch(j){

case 1:

staticVar++;

break;

case 2:

staticVar+=2;

case 4:

staticVar%=2;

j-=1;

continue;

default:

--staticVar;

continue;

}

}

printf("staticVar=%d",staticVar);

return 0;

}

Output: staticVar=-1

48.

#include <stdio.h>

#include <string.h>

int main() {

char a[10] = {'a','b','c','d'};

int l = strlen(a);

printf("%d",l);

return 0;

}

OUTPUT : 4

49.

// Online C compiler to run C program online

#include <stdio.h>

#include <string.h>

int main() {

int a,b;

for(a=6,b=4; a<=24; a=a+6){

if(a%b == 0){

break;

}

printf("%d",a);

}

return 0;

}

OUTPUT: 6

50.

// Online C compiler to run C program online

#include <stdio.h>

#include <string.h>

int main() {

static int staticVar;

int j;

for (j = 0; j <= 5; j += 2)

switch (j) {

case 1:

staticVar++;

break;

case 2:

staticVar += 2;

case 4:

staticVar %= 2;

j = -1;

continue;

default:

--staticVar;

continue;

}

printf("staticVar is:%d", staticVar);

return 0;

}

OUTPUT : staticVar is:0

51.

#include <stdio.h>

#include <string.h>

int main() {

static int staticVar;

int j;

for (j = 0; j <= 5; j += 2){

switch (j) {

case 1:

staticVar++;

break;

case 2:

staticVar += 2;

case 4:

staticVar %= 2;

j = -1;

continue;

default:

--staticVar;

continue;

}

printf("staticVar is:%d", staticVar);

}

return 0;

}

OUTPUT : StaticVar is : 2

Explanation: Since continue is present the print statement will not execute all times, only when break statement is executed the print statement will be executed.

52.