**1**. What would be the output of the following?

#include<stdio.h>

int main()

{

char ch[10\*1/2]={'T','e','s','t','\0'};

printf("%c%c%c",ch[2],ch[1],ch[3]);

return 0;

}

A. Character Array not valid, due to given size.

**B. set**

C. eTs

D. Error

**2.** What would be the output of the following?

#include<stdio.h>

int main()

{

char xyz[4][3]={"RAT","SAT","PAT","BAT"};

printf("%c",xyz[3][0]);

return 0;

}

A. P

**B. B**

C. T

D. Error

**3.** What would be the output of the following?

#include<stdio.h>

int main()

{

struct stud

{

int roll;

int age;

};

struct stud s1={101, 15};

struct stud s2;

s2=s1;

if(s1==s2)

printf("The structures are equal");

else

printf("The structures are not equal");

return 0;

}

}

A. The structures are equal

B. The structures are not equal

**C. Compile Time Error in code**

D. None

**4.** Predict the output of following code:

#include<stdio.h>

int main()

{

int list[5],i;

\*list=5;

for(i=1;i<=5;i++)

\*(list+i)=\*(list+i-1)\*i;

printf("\n");

for(i=1;i<=5;i++)

printf("%d ",\*(list+i));

return 0;

}

A. 5 10 15 20 25

**B. 5 10 30 120 600**

C. 5 5 5 5 5

D. Error

**5.** Predict the output of following code:

#include<stdio.h>

int main()

{

struct val

{

int net;

};

struct val x;

struct val \*p;

p=&x;

p->net=50;

printf("%d",++(x.net));

return 0;

}

A. 50

**B. 51**

C. Segmentation Fault

D. Error

**6.** Predict the output of following code:

#include<stdio.h>

#include<string.h>

int main()

{

char x[20]="gorilla";

char y[20]="giraffe";

strcat(x,y);

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

return 0;

}

A. gorilla gorilla

B. giraffe giraffe

C. giraffegorilla giraffe

D. **gorillagiraffe giraffe**

**7.** For the given structure, identify the invalid statements:

struct emp

{

int code;

float salary;

};

1.) struct emp E1={101, 15};

2.) struct emp E2;

3.) E2=E1;

4.) printf(“%s”,E1);

5.) printf(“%d %f”,E1.code, E1.salary);

6.) if(E1==E2)

printf("The structures are equal");

7.) if(E1.code==E2.code)

printf("Code Matched");

**A. 4 &6**

B. 3,4 &6

C. 1,3,4,&7

D. All are valid

**8.** The operator \* signifies a

A. Referencing operator

**B. Dereferenicng operator**

C. Address operator

D. None

**9.** Predict the output of following code:

#include<stdio.h>

void func(int(\*parr)[3]);

int main()

{

int arr[2][3]={1,2,3,4,5,6};

func(arr);

func(arr+1);

return 0;

}

void func(int (\*parr)[3])

{

int i;

for(i=0;i<2;i++)

printf("%d ",(\*parr)[i]);

}

A. 1 2 3 4

B. 2 3 5 6

**C. 1 2 4 5**

D. None

**10.** Predict the output of following code:

#include<stdio.h>

int main()

{

int arr[]={0,1,2,0,1,2,0,1,2};

printf("%d ",arr[3]);

printf("%d ",arr[arr[3]]);

printf("%d ",arr[arr[3]]+arr[1]);

printf("%d",arr[arr[arr[arr[1]]]]);

return 0;

}

A. 1 1 2 2

B. 0 1 2 1

C. 1 2 1 2

**D. 0 0 1 1**

**11.** Predict the output of following code:

#include<stdio.h>

int main()

{

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

int \*parr=arr+2;

printf("%d %d %d",\*parr,++\*parr-1,1+\*parr);

return 0;

}

A. 1 2 3

**B. 4 3 4**

C. 4 4 4

D. 4 3 3

**12.** Predict the output of given code snippet:

#include<stdio.h>

int main()

{

static int a[2][2] = {1, 2, 3, 4};

int i, j;

static int \*p[] = {(int\*)a, (int\*)a+1, (int\*)a+2};

for(i=0; i<2; i++)

{

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

{

printf("%d, %d, %d, %d\n", \*(\*(p+i)+j), \*(\*(j+p)+i),

\*(\*(i+p)+j), \*(\*(p+j)+i));

}

}

return 0;

}

A. 1, 2, 1, 2

2, 3, 2, 3

3, 4, 3, 4

4, 2, 4, 2

B. 1, 2, 3, 4

2, 3, 4, 1

3, 4, 1, 2

4, 1, 2, 3

C. 1, 1, 1, 1

2, 3, 2, 3

3, 2, 3, 2

4, 4, 4, 4

**D. 1, 1, 1, 1**

**2, 2, 2, 2**

**2, 2, 2, 2**

**3, 3, 3, 3**

**13.** What will be the output of the code:

#include<stdio.h>

int main()

{

int arr[5], i=0;

while(i<5)

arr[i]=++i;

for(i=0; i<5; i++)

printf("%d, ", arr[i]);

return 0;

}

**A. garbage value, 1, 2, 3, 4**

B. 1, 2, 3, 4, 5

C. 0, 1, 2, 3, 4

D. 1, 1, 1, 1, 1

**14.**  If a two dimensional array is used as a formal parameter, then

A. Both the subscripts may be left empty

**B. The first (row) subscript may be left empty**

C. The first subscript must be left empty

D. Both the subscripts must be left empty

**15.** Predict the output of following program, assume that a character takes 1 byte and pointer takes 4 bytes.

#include <stdio.h>

int main()

{

char \*str1 = "SWEETHOME";

char str2[] = "SWEETHOME";

printf("sizeof(str1) = %d, sizeof(str2) = %d",

sizeof(str1), sizeof(str2));

return 0;

}

A. sizeof(str1) = 10, sizeof(str2) = 10

**B. sizeof(str1) = 4, sizeof(str2) = 10**

C. sizeof(str1) = 10, sizeof(str2) = 4

D. sizeof(str1) = 4, sizeof(str2) = 4

**16.** What will be the output of the code:

#include <stdio.h>

void my\_toUpper(char\* str, int index)

{

\*(str + index) &= ~32;

}

int main()

{

char\* arr = "sweethome";

my\_toUpper(arr, 0);

my\_toUpper(arr, 5);

printf("%s", arr);

return 0;

}

A. SweetHome

B. sweethome

C. SWEETHOME

**D. Compiler Dependent or Segmentation fault**

**17.** What will be the output of the code:

#include <stdio.h>

struct p

{

int x;

char y;

};

int main()

{

struct p p1[] = {1, 92, 3, 94, 5, 96};

struct p \*ptr1 = p1;

int x = (sizeof(p1) / sizeof(ptr1));

if (x == 1)

printf("%d\n", ptr1->x);

else

printf("false\n");

return 0;

}

A. compile time error

B. 1

**C. false**

D. None

**18.** What will be the output of the code:

#include<stdio.h>

#include<string.h>

int main()

{

char p[] = "gorilla";

char t;

int i, j;

for(i=0,j=strlen(p); i<j; i++)

{

t = p[i];

p[i] = p[j-i];

p[j-i] = t;

}

printf("%s", p);

return 0;

}

**A. Nothing is printed on the screen**

B. ggggggg

C. aaaaaaa

D. allirog

**19.** What will be the output of the code:

#include <stdio.h>

void f(char\*\*);

int main()

{

char \*name[] = { "ab", "cd", "ef", "gh", "ij", "kl" };

f(name);

return 0;

}

void f(char \*\*p)

{

char \*t;

t = (p += sizeof(int))[-1];

printf("%s\n", t);

}

A. ab

B. cd

C. ef

**D. gh**

**20.** Which of the following is an incorrect syntax to pass by reference a member of a structure in a function?

(Assume: struct temp{int a;}s;)

A. func(&s.a);

B. func(&(s).a);

C. func(&(s.a));

**D. None**