Branch : CSE & IT Batch:Hinglish

WEEKLY TEST – 04

Subject : Programming in C

Topic : Recursion and arrays

Maximum Marks 20

Q.1 to 6 Carry ONE Mark Each

[MCQ]

1. Consider the following program:

```
#include <stdio.h>
int main()
{
  int b[5]={1, 2, 3, 4, 5};
  int *ptr[5]={b+3, b+2, b, b+1, b+4};
  int **q=ptr;
  *++*q;
  printf("%d\t%d\t%d", q-ptr, *q-b, **q);
  return 0;
}
```

The output of the program is-

(a) 0 1 2 (b) 0 4 5

(c) 1 2 3 (d) Compilation error.

[NAT]

2. Consider the following function:

```
void f(int n)
{
 if(n<2) {printf("%d", n-3); return;}
 printf("%d", n-2);
 f(n-1);
}
```

The sum of all the values printed when f(7) is called is _____.

[MCQ]

3. Consider the following program:

```
#include <stdio.h>
int main()
{
  int b[5]={1, 2, 3, 4, 5};
  int *ptr[5]={b+3, b+2, b, b+1, b+4};
  int **q=&ptr[3];
  --**q;
  printf("%d\t%d\t%d", q-ptr, *q-b, **q);
  return 0;
}
```

The output is-

(a) Compilation error

(b) 3 1 3

(c) 2 1 2

(d) 3 1 1

[MCQ]

4. Consider the following program:

```
#include <stdio.h>
int main()
{
   int b[3]={6, 12, 20};
   int *ptr[3]={b, b+2, b+1};
   printf("%d", *ptr[*ptr[1]-*ptr[2]-8]);
   return 0;
}
```

The output is:

(a) Segmentation Fault

(b) Compilation Error

(c) 6

(d) 12

[NAT]

5. Consider the following function:

```
int f(int x)
{
  if(x<=1) return x;
  if(x%3) return f(x/3)+x;
  return f(x/2)+x;
}
```

The value returned by f(24) is_____.

[MCQ]

6. Consider the following program:

```
#include <stdio.h>
int main()
{
   int a[][3][2]={0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11};
   printf("%u\t", a);
   printf("%u\t", **a+1);
   printf("%u\t", *(*(*a+2))+1);
```

```
    printf("%u\t", a+1);
    return 0;
}
```

Assume array index starts from 1000 and integer size is 2 bytes.

The output is-

(a) 1000 1002 5 1012
(b) 1000 1004 6 1012
(c) 1000 1008 7 1024
(d) 1000 1012 7 1024

Q.7 to 13 Carry TWO Mark Each

[NAT]

7. Consider the following program:

```
#include <stdio.h>
void func(int *q)
{
   int *p;
   p=q++;
   *q=*q-*p;
   *p=*p-1;
}
int main()
{
   int i;
   int a[][3]={0, 1, 2, 3, 4, 5, 6, 7, 8};
   for(i=0;i<2;i++)
   func(a[i]);
   return 0;
}
```

The sum of elements of the first two rows of the array is_____.

[NAT]

8. Consider the following program:

```
#include<stdio.h>
int f(int *arr, int n)
{
  int c;
  if(n<=1) return *arr-n;
  else if(*arr%5==0)
       return *(arr+1) - f(arr+1, n-1);
  else
       return *arr + f(arr+1, n-1);
}
int main(){
   int a[]={1, 2, 5, 10};
   printf("%d",f(a,sizeof(a)/sizeof(a[0])));
   return 0;
}
```

The output is_____.

[NAT]

9. Consider the following program:

```
#include<stdio.h>
int main()
{
  int a[]={7, 2, 5, 10, 13};
  int count=1, i=4;
  while(i){
  count=count+(*(a+i)-i);
  i--;
  }
  printf("%d", count);
  return 0;
}
```

The output is_____.

[MCQ]

10. Consider the following program:

```
#include <stdio.h>
void doSomething(int **p)
{
   **p++;
   printf("%d\t", **p);
}
int main()
{
   int b[5]={6, 12, 20, 13, 7};
   int *ptr[5]={b+4, b+3, b+2, b+1, b};
   int i;
   for(i=0;i<4; i++)
   doSomething(ptr+i);
   return 0;
}
```

The output is-

(a) 13 12 6 20 (b) 13 20 12 6
(c) 7 20 12 13 (d) 20 12 13 7

[MCQ]

11. Consider the following function:

```
int func(int n)
{
   static int i=1, j;
   if(n<0) return 0;
   j+=func(n-i);
   i+=2;
   return j+n;
}
```

The above function computes-

(a) x^2+2x+1 (b) 2^x+1
(c) 2^x-1 (d) 2^x

[NAT]

12. Consider the following function:

```
int func(int a)
{
   static int x;
   if(a>5) return a-x--;
   a=a+x++;
   return func(a)+x;
}
```

The value returned by func(4) is_____.

[MCQ]

13. Consider the following program:

```
#include <stdio.h>
void doSomething(int **p)
{
   printf("%u\t", p[1][1]);
   printf("%u\t", *p[1]);
   printf("%u\t", *(*(p+2)+1));
   printf("%u\t",**p+3);
   printf("%u",*(*p+3));
}
int main()
{
   int a[]={1, 2, 13, 7};
   int b[]={3, 9, 6};
   int c[]={4, 5, 7};
   int *arr[]={a, b, c};
  doSomething(arr);
  return 0;
}
```

The output is:

(a) 9 5 3 4 7 (b) 3 5 9 7 4
(c) 9 3 5 7 7 (d) 9 3 5 4 7

Answer Key

1. **(b)**
2. **(13)**
3. **(d)**
4. **(c)**
5. **(46)**
6. **(a)**
7. **(10)**
8. **(4)**
9. **(21)**
10. **(b)**
11. **(c)**
12. **(10)**
13. **(d)**

Hints and Solutions

1. (b)

1000	1002	1004	1006	1008
1	2	3	4	5

2000	2004	2008	2012	2016
~~1006~~1008	1004	1000	1002	1008

3000
2000

*++*q;//*++*2000 = *++1006 =*1008
printf("%d\t%d\t%d", q-ptr, *q-b, **q);
q-ptr= (2000-2000)/4 = 0/4 =0
*q-b = *2000-1000= (1008-1000)/2=4
q=2000=*1008=5

2. (13)

The printed values are:

5 4 3 2 1 0 -2

Sum: 13

3. (d)

1000	1002	1004	1006	1008
1	~~2~~ 1	3	4	5

2000	2004	2008	2012	2016
1006	1004	1000	1002	1008

3000
2012

--**q;//--**2012 = --*1002 =*1008
printf("%d\t%d\t%d", q-ptr, *q-b, **q);
q-ptr= (2012-2000)/4 = 12/4 =3
*q-b = *2012-1000= (1002-1000)/2=1
q=2012=*1002=1

4. (c)

1000	1002	1004
6	12	20

2000	2004	2008
1000	1004	1002

=*ptr[*ptr[1]-*ptr[2]-8]
=*ptr[*1004-*1002-8]
=*ptr[20-12-8]
=*ptr[0]
=*1000
=6

5. (46)

24%3 is 0. Return f(24/2)+24. Return 22+24 i.e 46
12%3 is 0. Return f(12/2)+12. Return 10+12 i.e 22
6%3 is 0. Return f(6/2)+6. Return 4+6 i.e 10
3%3 is 0. Return f(3/2)+3. Return 1+3 i.e 4
1%3 is 1. Return f(1/3)+1. Return 1

6. (a)

printf("%u\t", a);//It points to the 0^{th} 3D array.
printf("%u\t", **a+1); //It points to the 1^{st} element of the 0^{th} row of the 0^{th} 3D array. So, 1002 is printed.
printf("%u\t", *(*(*a+2))+1); //It is the 1^{st} element of the 2^{nd} row of the 0^{th} 3D array. So, 5 is printed.
printf("%u\t", a+1);//It points to the 1^{st} 3D array. So, 1012 is printed.

7. (10)

1000	1002	1004	1006
1	2	5	10

The given function func(*q)is called for the 0^{th} and 1^{st} row of the 2D array.
It decrements the 0^{th} element of a row by 1. It subtracts the value of the 0^{th} element from the 1^{st} element in any row. So, the two rows are-
-1 1 2
2 1 5
Sum: 10

8. (4)

sizeof(a)/sizeof(a[0]) gives the size of the array. Size of the array is 4.

f(0,4):

Line1: arr=100, n=4;

Line2: 4<=1->FALSE;

Line3:*arr i.e 1%5!=0, so else part is executed.

f(*arr+1, n-1)=f(102,3) =3

*arr+f(*arr+1, n-1)=1+3=4

return 4; //return 4 to main

f(102,3):

Line1: arr=102, n=3;

Line2: 3<=1->FALSE;

Line3:*arr i.e 2%5!=0, so else part is executed.

f(*arr+1, n-1)=f(104,2) =1

*arr+c=2+1 i.e 3

return 3; //return 3 to Line3 of f(100,4)

f(104,2):

Line1: arr=104, n=2;

Line2: 2<=1->FALSE;

Line3:*arr i.e 5%5==0, so else if part is executed.

f(*arr+1, n-1)=f(106,1)=9

*(arr+1)-c=(10-9) i.e 1

return 1; //return 1 to Line3 of f(102,3)

f(106,1):

Line1: arr=106, n=1;

Line2: 1<=1->TRUE; //Return *arr-n i.e. (10-1) i.e 9 to Line3 of f(104,2)

Output: 4

9. (21)

1000	1002	1004	1006	1008
7	2	5	10	13

count	1

```
while(4)
{
    count=count+(*(a+4)-4);//(*(1008)-4)=9;
    count=1+9=10
    i--;//i=3
}
while(3)
{
    count=count+(*(a+3)-3);//(*(1006)-3)=7;
    count=10+7=17
    i--;//i=2
}
while(2)
{
    count=count+(*(a+2)-2);//(*(1004)-2)=3;
    count=17+3=20
    i--;//i=1
}
while(1)
{
    count=count+(*(a+1)-1);//(*(1002)-1)=1;
    count=20+1=21
    i--;//i=1
}
printf("%d", count);//21 is printed.
```

10. (b)

1000	1002	1004	1006	1008
6	12	20	13	7

2000	2004	2008	2012	2016
1008	1006	1004	1002	1000

For i=0:

doSomething(2000+0):

p=2000

**p++;//p=2004

printf("%d\t", **p);//**2004= 13 is printed.

For i=1:

doSomething(2000+1):

p=2004

**p++;//p=2008

printf("%d\t", **p);//**2008= 20 is printed.

For i=2:

doSomething(2000+2):

p=2008

**p++;//p=2012

printf("%d\t", **p);//**2012= 12 is printed.

For i=3:

doSomething(2000+3):

p=2012

**p++;//p=2016

printf("%d\t", **p);//**2016= 6 is printed.

Output: 13 20 12 6

11. (c)

The above function computes 2^x-1.

12. (10)

func(4):

a=a+x++; //a=4+0=4. Static x is incremented to 1.

return func(4)+x; return 10

func(4):

a=a+x++; //a=4+1=5. Static x is incremented to 2.

return func(5)+x; return 6+2; return 8

func(5):

a=a+x++; //a=5+2=7. Static x is incremented to 3.

return func(7)+x; return 6

func(7):

return a-x--;

return 7-3=4; x is decremented to 2.

13. (d)

1000	1002	1004	1006
1	2	13	7

2000	2002	2004
3	9	6

3000	3002	3004
4	5	7

4000	4004	4008

1000	2000	3000

doSomething(4000):

p	4000

printf("%u\t", p[1][1]);
//*(*(p+1)+1)=*(*(4000+1)+1)=*(*4004+1)=*(2000+1)=*2002=9
printf("%u\t", *p[1]);//**(p+1)=**(4000+1)=**4004=*2000=3
printf("%u\t", *(*(p+2)+1));= *(*(4000+2)+1)= *(*4008+1)=*(3000+1)=*3002=5
printf("%u\t",**p+3);=**4000+3=*1000+3=1+3=4
printf("%u",*(*p+3));=*(*4000+3)=*(1000+3)=*1006=7
Output: 9 3 5 4 7