RECURSION

Solutions

1. Consider the following recursive C function. If get(6) function is being called in main() then how many times will the get() function be invoked before returning to the main()?

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

(a) 15 (b) 25
(c) 35 (d) 45

Solution: Option (b)

2. Consider the following recursive C function that takes two arguments:

```
unsigned int foo(unsigned int n, unsigned int r) {
 if (n  > 0) return (n%r +  foo (n/r, r ));
else return 0;
}
```

What is the return value of the function foo when it is called as foo(345, 10)?

(a) 345 (b) 12
(c) 5 (d) 3

Solution: Option (b)

Explanation:
The call foo(345, 10) returns sum of decimal digits (because r is 10) in the number n. Sum of digits for 345 is 3 + 4 + 5 = 12.

3. Consider the same recursive C function that takes two arguments:

```
unsigned int foo(unsigned int n, unsigned int r) {
 if (n  > 0) return (n%r +  foo (n/r, r ));
else return 0;
}
```

What is the return value of the function foo when it is called as foo(513, 2)?

(a) 9	(b) 8
(c) 5	(d) 2

Solution: Option (d)

Explanation:
foo(513, 2) will return 1 + foo(256, 2). All subsequent recursive calls (including foo(256, 2)) will return 0 + foo(n/2, 2) except the last call foo(1, 2) . The last call foo(1, 2) returns 1. So, the value returned by foo(513, 2) is 1 + 0 + 0…. + 0 + 1.

The function foo(n, 2) basically returns sum of bits (or count of set bits) in the number n.

4.

```
#include<stdio.h>
int f(int *a, int n)
{
if(n ≤ 0) return 0;
else if(*a % 2 = = 0) return *a + f(a+1, n-1);
else return *a - f(a+1, n-1);
}

int main()
{
  int a[] = {12, 7, 13, 4, 11, 6};
  printf("%d", f(a, 6));
  getchar();
  return 0;
}
```

(a) -9	(b) 5
(c) 15	(d) 19

Solution: Option (c)

Explanation:
f() is a recursive function which adds f(a+1, n-1) to *a if *a is even. If *a is odd then f() subtracts f(a+1, n-1) from *a.

5. Output of following program?

```
#include <stdio.h>
int fun(int n, int *f_p)
{
   int t, f;
   if (n ≤ 1)
   {
        *f_p = 1;
        return 1;
   }
   t = fun(n- 1,f_p);
   f = t + * f_p;
   *f_p = t;
   return f;
}

int main()
{
   int x = 15;
   printf (" %d \n", fun(5, &x));
   return 0;
}
```

(a) 6 (b) 8
(c) 14 (d) 15

Solution: Option (b)

6. Consider the following function:

```
double f(double x){
if( abs(x*x – 3) < 0.01) return x;
else return f(x/2 + 1.5/x);
}
```

Give a value q (to 2 decimals) such that f(q) will return q:________.

(a) 1.72 (b) 2.24
(c) 4.22 (d) 3.42

Solution: Option (a)

7. Consider the C function given below:

```
int f(int j)
{
static int i = 50;
int k;
if (i = = j)
 {
  printf("something");
  k = f(i);
  return 0;
  }
else return 0;
}
```

Which one of the following is TRUE?

(a) The function returns 0 for all values of j.
(b) The function prints the string something for all values of j.
(c) The function returns 0 when j = 50.
(d) The function will exhaust the runtime stack or run into an infinite loop when j = 50.

Solution: Option (d)

Explanation:
When j is 50, the function would call itself again and again as neither i nor j is changed inside the recursion.

8.

```
#include<stdio.h>
void crazy(int n, int a, int b)
{
 if (n ≤ 0)  return;
crazy(n-1, a, b + n);
printf("%d %d %d\n", n, a, b);
crazy(n-1, b, a + n);
}

int main()
{
   crazy(3, 4, 5);
```

```
    return 0;
}
```

(a) 1 4 10
    2 4 8
    1 8 6
    3 4 5
    1 5 9
    2 5 7
    1 7 7

(b) 3 4 5
    1 4 10
    2 4 8
    1 8 6
    1 5 9
    2 5 7
    1 7 7

(c) 1 4 10
    2 4 8
    1 8 6
    3 4 5

(d) 3 4 5
    1 5 9
    2 5 7
    1 7 7

Solution: Option (a)

9. Consider the following C function:

```
int f(int n)
{
static int i = 1;
if (n ≥ 5)
  return n;
n = n + i;
i++;
return f(n);
}
```

The value returned by f(1) is:

(a) 5 (b) 6
(c) 7 (d) 8

Solution: Option (c)

10. Consider the following C function:

```
int fun (int n)
{
```

```
int x=1, k;
if (n= =1) return x;
for (k=1; k < n; ++k)
     x = x + fun(k) * fun(n – k);
}
```

The return value of fun(5) is __________.

(a) 0 (b) 26
(c) 51 (d) 71

Solution: Option (c)

Explanation:
fun(5) = 1 + fun(1) * fun(4) + fun(2) * fun(3) + fun(3) * fun(2) + fun(4) * fun(1)
= 1 + 2*[fun(1)*fun(4) + fun(2)*fun(3)]

Substituting fun(1) = 1
= 1 + 2*[fun(4) + fun(2)*fun(3)]

Calculating fun(2), fun(3) and fun(4):
fun(2) = 1 + fun(1)*fun(1) = 1 + 1*1 = 2
fun(3) = 1 + 2*fun(1)*fun(2) = 1 + 2*1*2 = 5
fun(4) = 1 + 2*fun(1)*fun(3) + fun(2)*fun(2)
= 1 + 2*1*5 + 2*2 = 15

Substituting values of fun(2), fun(3) and fun(4):
fun(5) = 1 + 2*[15 + 2*5] = 51

11. Predict output of following program:

```
#include <stdio.h>

int fun(int n)
{
if (n = = 4)
      return n;
else return 2*fun(n+1);
}

int main()
{
printf("%d ", fun(2));
return 0;
}
```

(a) 4 (b) 8
(c) 16 (d) Runtime Error

Solution: Option (c)

12. Consider the following recursive function fun(x, y). What is the value of fun(4, 3)?

```
int fun(int x, int y)
{
  if (x = = 0)
  return y;
return fun(x - 1,  x + y);
}
```

(a) 13 (b) 12
(c) 9 (d) 10

Solution: Option (a)

Explanation:
The function fun() calculates and returns ((1 + 2 … + x-1 + x) + y) which is x(x+1)/2 + y.

13. What does the following function print for n = 25?

```
void fun(int n)
{
if (n = = 0)
  return;
printf("%d", n%2);
fun(n/2);
}
```

(a) 11001 (b) 10011
(c) 11111 (d) 00000

Solution: Option (b)

Explanation:
The function mainly prints binary representation in reverse order.

14. What does the following function do?

```
int fun(int x, int y)
{
if (y = = 0)   return 0;
return (x + fun(x, y-1));
}
```

(a) x + y
(b) x + x*y
(c) x*y
(d) x^y

Solution: Option (c)

Explanation:
The function adds x to itself y times which is x*y.

15. What does fun2() do in general?

```
int fun(int x, int y)
{
  if (y = = 0)   return 0;
  return (x + fun(x, y-1));
}

int fun2(int a, int b)
{
  if (b = = 0) return 1;
  return fun(a, fun2(a, b-1));
}
```

(a) x*y
(b) x+x*y
(c) x^y
(d) y^x

Solution: Option (c)

Explanation:
The function multiplies x to itself y times which is xy.

16. Output of following program?

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

```
void print(int n)
{
   if (n > 4000)
   return;
printf("%d ", n);
print(2*n);
printf("%d ", n);
}

int main()
{
  print(1000);
  getchar();
return 0;
}
```

(a) 1000 2000 4000	(b) 1000 2000 4000 4000 2000 1000
(c) 1000 2000 4000 2000 1000	(d) 1000 2000 2000 1000

Solution: Option (b)

17. What does the following function do?

```
int fun(unsigned int n)
{
if (n = = 0 || n = = 1)
  return n;

if (n%3 != 0)
  return 0;
  return fun(n/3);
}
```

(a) It returns 1 when n is a multiple of 3, otherwise returns 0
(b) It returns 1 when n is a power of 3, otherwise returns 0
(c) It returns 0 when n is a multiple of 3, otherwise returns 1
(d) It returns 0 when n is a power of 3, otherwise returns 1

Solution: Option (b)

18. Predict the output of following program:

```
#include <stdio.h>
int f(int n)
{
if(n ≤ 1)
        return 1;
if(n%2 = = 0)
        return f(n/2);
return f(n/2) + f(n/2+1);
}

int main()
{
printf("%d", f(11));
return 0;
}
```

(a) Stack Overflow (b) 3
(c) 4 (d) 5

Solution: Option (d)

19. Which of the following operations is not O(1) for an array of sorted data. You may assume that array elements are distinct.

(a) Find the ith largest element
(b) Delete an element
(c) Find the ith smallest element
(d) All of the above

Solution: Option (b)

Explanation:
The worst case time complexity for deleting an element from array can become O(n).

20. A program P reads in 500 integers in the range [0..100] representing the scores of 500 students. It then prints the frequency of each score above 50. What would be the best way for P to store the frequencies? (GATE CS 2005)

(a) An array of 50 numbers
(b) An array of 100 numbers
(c) An array of 500 numbers
(d) A dynamically allocated array of 550 numbers

Solution: Option (a)