DATA STRUCTURE & ALGORITHM

1. The minimum number of temporary variables needed to swap the contents of two variables is:

(a) 1 (b) 2
(c) 3 (d) 0

2. Consider the function:

```
find (int x, int y)
{ return((x < y) ? 0: (x – y)); }
```

Let a, b be two non-negative integers. The call find(a, find(a, b)) can be used to find the:
(a) maximum of a, b (b) positive difference of a, b
(c) sum of a, b (d) minimum of a, b

3. The following:

```
printf (“%f”, 9/5);
```

prints:
(a) 1.8 (b) 1.0
(c) 2.0 (d) none of the above

4. If an integer needs two bytes of storage then maximum value of unsigned integer is:

(a) $2^{16}–1$ (b) $2^{15}–1$
(c) 2^{16} (d) 2^{15}

5. If an integer needs two bytes of storage then maximum value of a signed integer is:

(a) $2^{16}–1$ (b) $2^{15}–1$
(c) 2^{16} (d) 2^{15}

6. printf(“%d”, printf(“tim”));

(a) results in a syntax error
(b) outputs tim3
(c) outputs garbage
(d) prints tim and terminates abruptly

7. If a b c is the input then the following program fragment results in:

```
char x, y, z;
printf("%d", scanf("%c %c %c", &x, &y, &z));
```

results in:

(a) a syntax error
(b) a fatal error
(c) segmentation violation
(d) printing of 3

8. Consider the statements:

```
putchar(getchar( ) );
putchar(getchar( ) );
```

if a
   b
is the input, the output will be:

(a) an error message
(b) this can't be the input
(c) ab
(d) a b

9. Let a, b be two positive integers, which of the following options correctly relates / and %?

(a) b= (a/b) * b + a%b
(b) b= (a%b) * b + a/b
(c) a= (a/b) * b + a%b
(d) a= (a%b) * b + a/b

10. Consider the following program fragment:

```
char c= 'a'
while (c++ ≤ 'z')
putchar (xxx);
```

if the required output is abcdefghijklmnopqrstuvwxyz then xxx should be:

(a) c
(b) c++
(c) c–1
(d) –c

11. If y is of integer type then the expressions:
3 * (y–8)/9 and (y–8)/9 * 3

(a) must yield same value
(b) must yield different values
(c) may or may not yield the same value
(d) none of the above

12. The statement:

```
if (my Ptr != NULL)
   *myPtr= NULL;
else
   *myPtr= NULL;
```

has the same effect as the statement(s):

(a) if (myPtr) * myPtr= NULL;
      else *myPtr= NULL;
(b) *myPtr= NULL;
(c) if (!myPtr) *myPtr= NULL;
      else *myPtr= NULL;
(d) All of the above

13. The following code fragment:

```
int x, y= 2, z, a;
x= (y* =2) + (z= a =y);
printf("%d", x);
```

(a) prints 8
(b) prints 6
(c) prints 6 or 8 depending on the compiler implementation
(d) is syntactically wrong

14. If n has the value 3 then the output of statement:

```
printf("%d %d", n++, ++n);
```

(a) is 3 5
(b) is 4 5
(c) is 4 4
(d) is implementation dependent

15. x– = y+1; does the same as:

(a) x= x –y +1 (b) x= – x –y – 1
(c) x= –x + y +1 (d) x= x – y – 1

16. The expression 5 – 2 – 3 * 5 – 2 will evaluate to 18, if:

(a) – is left associative and * has precedence over –
(b) – is right associative and * has precedence over –
(c) – is right associative and – has precedence over *
(d) – is left associative and – has precedence over *

17. printf (“%c”, 100);

(a) prints 100 (b) prints the ASCII equivalent of 100
(c) prints garbage (d) none of the above

18. The program fragment:

```
int i= 263;
putchar (i)
```

(a) prints 263 (b) prints the ASCII equivalent of 263
(c) rings the bell (d) prints garbage