

## SECTION A True or False

Use the answer sheet provided for answering the questions in this section. Each correct answer will garner 1 mark.

1. `_variable1` is an invalid C identifier.

- (a) True
- (b) False

Answer: (b) False

2. `6.022E23L` is a valid floating point literal.

- (a) True
- (b) False

Answer: (a) True

3. The statement `int c = 'A'++;` is valid causing the variable `c` to have a value of 66 since the numeric value of the character `'A'` is 65.

- (a) True
- (b) False

Answer: (b) False

4. The expression `5.5 + 'X' / 8` evaluates to a value which has type `float`.

- (a) True
- (b) False

Answer: (b) False

5. The base address of an array is the address of the first array component.

- (a) True
- (b) False

Answer: (a) True

6. The following C code will compile without errors:

```
int foo(const int *a, const int *b)
{
    (*b)++;
    return *a + *b;
}
```

- (a) True
- (b) False

Answer: (b) False

7. When executed, the following C program will complete without any issues:

```
#include <stdio.h>

int main(void)
{
    char *str = "Hello";
    str[0] = 'h';
}
```

(a) True

(b) False

**Answer: (b) False**

8. The operand of the indirection operator can be a variable of any type.

(a) True

(b) False

**Answer: (b) False**

9. In the following declaration

```
register int i;
```

the value of variable `i` is guaranteed to be stored in a CPU register.

(a) True

(b) False

**Answer: (b) False**

10. Declaring auto variables of the same name in two different non-overlapping blocks will not cause compilation issues.

(a) True

(b) False

**Answer: (a) True**

11. Identifiers in an enumeration declaration can be explicitly assigned floating point values.

(a) True

(b) False

**Answer: (b) False**

12. Consider the following code snippet:

```
char *ptr = (char *)malloc(8*sizeof(char));  
realloc(ptr, 12*sizeof(char));
```

After the call to `realloc()` on the second line, `ptr` still points to the previously allocated memory on the the first line.

- (a) True
- (b) False

**Answer: (a) True**

13. A singly-linked list can only be traversed in one direction starting from the head.

- (a) True
- (b) False

**Answer: (a) True**

14. When a program reads from the `stdin` stream which is connected to the keyboard, the program immediately receives every character inputted by the user.

- (a) True
- (b) False

**Answer: (b) False**

15. In the call `fflush(fp)`, `fp` must be a stream opened either in write or append mode.

- (a) True
- (b) False

**Answer: (a) True**

## SECTION B Multiple Choice

Use the answer sheet provided for answering the questions in this section.

Each correct answer will garner 1 mark.

16. Which of the following is an **invalid** integer literal?

- (a) 1234
- (b) 0xbeef
- (c) -100U
- (d) 0239

Answer: (d)

17. A C program contains the following declarations:

```
int i, j;  
long ix ;  
short s;  
float x;  
char c;
```

What is the resulting data type of the expression?

```
3.5 * i + (short) (ix / s) - x * c / j
```

- (a) float
- (b) double
- (c) int
- (d) long int

Answer: (b)

18. Consider the following function-like macro:

```
#define FUNCMACRO(X,Y) X/Y
```

What value does the macro evaluate when invoked as `FUNCMACRO(1+8, 4-3)`?

- (a) 0
- (b) 9
- (c) the string "1+8/4-3"
- (d) None of the above

Answer: (a)

19. Consider the following statement:

```
char str[] = "Seven";
```

What is the size of the array `str`?

- (a) 5
- (b) 6
- (c) 7
- (d) None of the above

**Answer: (b)**

20. Consider the following C code snippet:

```
char str1[] = "String 1";  
char *str2 = "String 2";
```

Select ALL valid statements from the following:

- i. `str1[0] = 's';`
- ii. `str2[0] = 's';`
- iii. `strcpy(str1, str2);`
- iv. `strcpy(str2, str1);`
- v. `str2 = str1;`

- (a) i and iii
- (b) ii and iv
- (c) i, iii and v
- (d) ii, iv and v

**Answer: (c)**

21. Suppose the following declarations are given:

```
int i = 5, j = 10, *ip;  
ip = &i;
```

Which of the following statements use `*` for indirection?

- (a) `int *x = ip;`
- (b) `i = i*j;`
- (c) `j = j**ip;`
- (d) `int **y = &ip;`

**Answer: (c)**

22. Consider the following code snippet:

```
int a = 2, b = 3, *x, *y;  
x = &a;  
y = &b;  
*x = *x + *y;
```

What is the resulting value of a?

- (a) 2
- (b) 3
- (c) 5
- (d) 8

**Answer: (c)**

23. Consider the following C snippet:

```
int a[] = {2, 4, 6, 8};  
int *p = a;
```

Select ALL expressions that will return the value of the third element of the array a, that is, the value 6.

- i. a[2]
- ii. \*a+2
- iii. \*(p+2)
- iv. p[2]
- v. p+2

- (a) i and iv
- (b) i, iii, and iv
- (c) i, ii, and v
- (d) i, iv, and v

**Answer: (b)**

24. Consider the following C code snippet:

```
int n[] = {1, 2, 3, 4, 5, 6, 7, 8};  
int *p = n + *n;
```

What is the value of \*(n + \*p)?

- (a) 2
- (b) 3
- (c) 4
- (d) 5

**Answer: (b)**

25. Which of the following is equivalent to the call `malloc(10*sizeof(double))` ?

- (a) `calloc(10*sizeof(double))`
- (b) `calloc(10)`
- (c) `calloc(sizeof(double))`
- (d) `calloc(10, sizeof(double))`

**Answer: (d)**

26. Select ALL valid statements about memory leak from the following statements:

- i. Program will not be able to access leaked memory.
- ii. Leaked memory will no longer be in the heap segment.
- iii. Leaked memory cannot be freed, potentially causing program memory usage to keep on growing.
- iv. Leaked memory is automatically freed using garbage collection.
- v. Every instance of memory leak will always result in undefined program behaviour.

- (a) i
- (b) i and ii
- (c) i and iii
- (d) i, iii, and v

**Answer: (c)**

27. Consider the following C code snippet:

```
enum loudness { moderate, deafening = 2, painful };
```

What is the value of `painful`?

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

**Answer: (d)**

28. Consider the following code snippet:

```
union {  
    char c;  
    short s;  
    int i;  
    long l;  
} u;
```

```
u.c = 'A';
```

What is the size of the variable `u` equal to?

- (a) `sizeof(char)`
- (b) `sizeof(short)`
- (c) `sizeof(int)`
- (d) `sizeof(long)`

Answer: (d)

29. Which stream buffering mode is used if reading or writing occurs in arbitrarily sized blocks of characters?

- (a) Unbuffered
- (b) Line buffered
- (c) Fully buffered
- (d) Free buffered

Answer: (c)

30. Select ALL valid reasons for a file opening failure.

- i. File is already opened.
- ii. File opened for writing or append does not exist.
- iii. File is empty.
- iv. File cannot be accessed due to insufficient permissions.
- v. File is already closed.

- (a) i and ii
- (b) i and iv
- (c) ii and iii
- (d) iii and v

Answer: (b)



## SECTION C Short Answer

Write your answer in the space provided.

31. Consider the following declaration: **(2 marks)**

```
struct point {
    int x;
    int y;
};
```

Write a single statement declaring a variable `p1` of type `struct point` with the members `x` and `y` initialised to 10 and 20, respectively.

Any of the following:

```
struct point p1 = {10, 20};
```

```
struct point p1 = { x: 10, y: 20}; // x and y can be in any order
```

```
struct point p1 = { .x = 10, .y = 20}; // x and y can be in any order
```

32. Consider the following C program: **(1 mark)**

```
#include <stdio.h>

int foo(int a, int b)
{
    return ++b / a;
}

int main(void)
{
    int i = 4;
    int j = 2 * foo(1+2, i+1);
    printf("%d %d", i, j);
    return 0;
}
```

What is the output of the program?

4 4

33. Re-write `foo(int a, int b)` in program in the previous question into a function-like macro `F00(A, B)`, such that when the call to `foo(1+2, i+1)` in the program is replaced with `F00(1+2, i+1)`, the outputs will remain the same. **(2 marks)**

```
#define FOO(A,B) (((B)+1)/(A))
```

34. Given the following variable declarations:

```
short a[] = {1, 2, 3, 4, 5, 6};  
short *ip = a;
```

Suppose that a `short` occupies 2 bytes in memory. The array `a` is at memory address 100, while `ip` is at memory address 200 (all addresses are in decimal).

(a) What is the numeric value of the expression `a`? **(1 mark)**

100

(b) What is the numeric value of the expression `ip+1`? **(1 mark)**

$100+1*2 = 102$

(c) What is the numeric value of the expression `&a[2]`? **(1 mark)**

$100+2*2 = 104$

(d) What is the numeric value of the expression `*(ip+2)`? **(1 mark)**

3

(e) What is the numeric value of the expression `***ip`? **(1 mark)**

2

35. Consider the following C program:

```
1  #include<stdio.h>  
2  
3  int a;  
4  
5  int func(int i)
```

```

6  {
7      int b;
8      static int c = 10;
9      b = c;
10     if(i == 0) c = c+b;
11     else if(i < 0) c--;
12     else c++;
13
14     return c;
15 }
16
17 int main(void)
18 {
19     int d = -1, e;
20     func(d);
21     d++;
22     func(d);
23     e = func(++d);
24     printf("%d", e);
25     return 0;
26 }

```

(a) What is storage class of variable a? (1 mark)

extern

(b) In which memory segment is the variable b stored? (1 mark)

stack

(c) What is the lifetime of variable c? (1 mark)

static

(d) Until what line is variable e allocated space? (1 mark)

Until line 26 or last line of the program.

(e) What is the output of the program? (1 mark)

19

\*\*\*\*\*