UNIVERSITY OF BRISTOL

December 2024

School of Computer Science

MINI MOCK theory examination for the Degree of Master of Science in Computer Science (conversion)

COMSM1201
Programming in C

TIME ALLOWED:
45 minutes

This paper contains **twenty-five** questions.
All questions will be marked.
All questions must be answered on the multiple choice answer sheet.
The maximum for this paper is **25 marks**.

Other Instructions

Calculators and notes are not permitted for this examination.

All questions in this examination will be computer marked, so it is crucial that you fill in the answer sheet carefully. Use a pencil (not pen) to fill in the answer sheet, and a pencil eraser to remove any mistakes you make.

Write your candidate name, title of the examination, unit code, date, desk number, and your student number on the answer sheet in the relevant boxes.

TURN OVER ONLY WHEN TOLD TO BEGIN WORK

1. Which of the following is not a valid numeric constant?

   a) `7` b) `7.0` c) `07` d) `0o7` e) `0x7`

2. Which header file must be included to use the pre-defined `rand()` function?

   a) stdio.h b) stdlib.h c) stdbool.h d) string.h e) math.h

3. Which of the following is not a standard data type in C?

   a) signed
   b) signed char
   c) signed double
   d) signed long
   e) signed short

4. What does the static keyword do when applied to a local variable in a function?

   a) The variable retains its value between function calls.
   b) The variable is shared across all functions in the program.
   c) The variable cannot be modified once assigned a value.
   d) The variable is initialized to zero every time the function is called.
   e) The variable is accessible outside the file it is defined in.

5. For the following code snippet, what will be printed to the terminal?

```
enum tas {alex, ali=3, andrew, harry, phoenix, liz};
printf("%d", andrew);
```

   a) 0 b) 1 c) 2 d) 3 e) 4

6. What compiler flag is required in order to use the pre-defined functions defined in math.h?

   a) `-g3` b) `-lm` c) `-o` d) `-O3` e) `-std=99`

7. For the following code snippet, what will be printed to the terminal?

```
printf("%d\n", 3^2);
```

   a) 0 b) 1 c) 2 d) 6 e) 9

8. What will happen when attempting to compile and run a program that contains the following code snippet?

```
int i = 0, sum = 0;
while (i < 5) {
   sum += i;
}
printf("%d\n", sum);
```

   a) `0` will be printed to the terminal
   b) `5` will be printed to the terminal
   c) `10` will be printed to the terminal
   d) The code will not compile due to a syntax error
   e) The program will get stuck in an infinite loop

9. For the following code snippet, what will be printed to the terminal?

```
int a = 2;
int b = 3;
printf("%d\n", a = b);
```

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

10. What will the value of `l` be after the following snippet is executed?

```
int i = 2;
int j = 3;
int k = 6;
int l = (i * j > k) ? 5 : 4;
```

a) 2 b) 3 c) 4 d) 5 e) 6

11. Which of the following statements will not result in `fopen("example.txt", "r")` returning a NULL pointer?

    a) The file example.txt does not exist.
    b) The file example.txt exists, but the program lacks read permissions.
    c) The file example.txt exists, but is in a different directory to the program.
    d) The file example.txt is locked or occupied by another process.
    e) The file example.txt is empty.

12. For a program with a main function with the top line `int main(int argc, char *argv[])`, what does `argc` represent?

    a) The number of command-line arguments passed, including the program name.
    b) The number of characters in the command-line arguments.
    c) The index of the first command-line argument.
    d) The total number of bytes allocated for the command-line arguments.
    e) The length of the longest command-line argument.

13. Which of the following code snippets might result in different behaviour to the others, when followed by the line `printf("%s\n", x);`?

    a) `char* x = "x"`
    b) `char x[3] = "x";`
    c)
    ```
    char x[3];
    *x = 'x';
    *(x+1) = '\0';
    ```
    d)
    ```
    char *x = (char*) malloc(3*sizeof(char));
    x[0] = 'x';
    ```
    e)
    ```
    char* x = (char*) calloc(3,sizeof(char));
    *x = 'x';
    ```

14. For the following code snippet, what will be printed to the terminal?

```
int func(const int* a, int n){
   if(n >= 0){
      return a[a[n]];
   }
   return 0;
}

int main(void){
   int a[] = {1,2,3,4};
   printf("%d\n", func(a, 2));
   return 0;
}
```

a) `0` b) `1` c) `2` d) `3` e) `4`

15. For the following code snippet, what will be printed to the terminal?

```
int a = 1, b = 2;
a = b, b = a--;
printf("%d", --a);
a = b--;
printf("%d", a/b);
```

a) `01`
b) `02`
c) `11`
d) `12`
e) `Floating point exception (core dumped)`

16. What error will occur when attempting to compile and run code that contains the following snippet, if the program is compiled using the sanitizer flags?

```
int func(void){
   int* a = calloc(5, sizeof(int));
   int b = 10;
   for (int i = 0; i <= 5; i++){
      a[i] = i - 1;
   }
   free(a);
   return b;
}

int main(void){
   int a = 5;
   int b = func();
   int c = a + b;
   return 0;
}
```

a) stack-buffer-overflow
b) heap-use-after-free
c) heap-buffer-overflow
d) Compilation error
e) No error will occur

17. In which order should the pointers in the below code snippet be freed?

```
node1->next = node2;
node2->next = node3;
```

a) `node1, node2, node3`
b) `node1, node3, node2`
c) `node2, node3, node1`
d) `node3, node1, node2`
e) `node3, node2, node1`

18. For the below code snippet, which of the following function calls would allow the value of `x` to be edited?

```
int y = 5;
int* x = &y;
```

a) `func(&x);` where the `func()` prototype is `void func(int* a);`
b) `func(&y);` where the `func()` prototype is `void func(int* a);`
c) `func(*x);` where the `func()` prototype is `void func(int* a);`
d) `func(&x);` where the `func()` prototype is `void func(int** a);`
e) `func(&y);` where the `func()` prototype is `void func(int** a);`

19. Given the below function definitions and a tree with root node `rt`, if `func1(rt)` prints `ABDECF` and `func2(rt)` prints `DBEAFC`, what will `func3(rt)` print?

```
void func1(node* n) {
    if (n != NULL) {
        printf("%c", n->data);
        func1(n->left);
        func1(n->right);
    }
}

void func2(node* n) {
    if (n != NULL) {
        func2(n->left);
        printf("%c", n->data);
        func2(n->right);
    }
}

void func3(node* root) {
    if (n != NULL) {
        func3(n->left);
        func3(n->right);
        printf("%c", n->data);
    }
}
```

a) `DBEFCA` b) `DEBFCA` c) `EBDFCA` d) `EDBFAC` e) `EDBFCA`

20. What is the average time complexity of inserting an element at the end of a fixed size array, if there's space available?

    a) O(n) b) O(log log n) c) O(1) d) O(log n) e) O(n^2)

21. What is the average time complexity of an binary search on a sorted array?

    a) O(n) b) O(log log n) c) O(1) d) O(log n) e) O(n^2)

22. Which of the following do users of an Abstract Data Type (ADT) need knowledge of?

    a) The data type(s) used to store the ADT's data.
    b) The operations that can be performed on the ADT's data.
    c) The ordering of data being stored in the ADT.
    d) The variable names used to store data in the ADT.
    e) The source code written to implement the ADT.

23. Which of the following data structures would be preferable to store a list of strings, if speed of searching is the top priority?

    a) Stack b) Array c) Linked list d) Binary tree e) Hash table

24. Which of these ADTs doesn't allow duplicate elements to be added?

    a) Stacks b) Queues c) Sets d) Graphs e) Trees

25. Which of the following statements best describes a hashing algorithm?

    a) A function that maps input data to a fixed-size value to facilitate efficient data retrieval.
    b) A method for encrypting data to prevent unauthorized access.
    c) A process of compressing data into a smaller fixed-size format for storage efficiency.
    d) A function that generates a random value for each piece of input data.
    e) A procedure for iteratively improving the accuracy of search results.

THIS PAGE IS INTENTIONALLY LEFT BLANK

THIS PAGE IS INTENTIONALLY LEFT BLANK