Fundamentals of Computing and Programming

Mid Term Examination. 50 marks, Time Limit 2 hrs

October 14, 2022

1. **Rewrite the code below** using a single **switch** instead of using **if**: [3]

```
void f(int x){
  if (x == 2 )
       f();
  if ( x == 2 || x == 3 )
       common();
  else
         h();
}
```

2. **Rewrite this code below** using **for** instead of `do ... while`: [3]

```
void g(int i){
    do {
        printf( "%d \n", i);
        i--;
    } while ( i > 0 );
}
```

3. This question is with respect to a function called `quiz`:

   (a) **Declare the prototype** of the function given it takes **two parameters**- the first is a character `c` and the second is an integer `n`. It **returns a character**. [1]

(b) The body contains **exactly one line**, it returns the character that is `n` positions after the character in `c`. **Write out the function**. [2]

4. **Write a function** named `myrange` to take two integer **parameter** values x and y. It prints all integers greater than x but less than y in decreasing order. It normally returns 0; however, if the range is empty, it returns -1. [3]

5. Answer in one line for each of these: [8]

   (a) Differentiate C source code from a compiled executable code.

   (b) What does a `#include` directive do?

   (c) What is the difference between the directive `#define x 2` and the statement `int x 2;`?

   (d) How does a parameter to a function get its value?

   (e) To be useful a recursive function always needs a condition check. Why?

   (f) Why do we need the ampersand sign in this use of `scanf`:
   ```
   int x; scanf("%d",&x)
   ```

   (g) What does the following print and why:
   ```
   float f = 5 / 2 ;  printf("%f", f)
   ```

   (h) Assuming `a` and `b` are integer variables, which are already initialized, what do the following statements achieve:

   ```
   a=a+b;
   b=a-b;
   a=a-b;
   ```

6. What does the following code print: [3]

```
if (2 < 1 < 3)
    printf("1: True\n");
if (1 > 0 == 0)
    printf("2: True\n");
if (1 > 2 <= 0)
    printf("3: True\n");
```

7. Assume there is a function `random1000()` which returns a random integer in the range $[0 \ldots 1000]$. **Write a while loop** that repeatedly iterates as long as `random1000()` returns a value less than 500. Inside the loop it reads another integer from the user. If that integer is negative it immediately terminates the loop. [3]

8. **Write a function** `void swap(int *p1, int *p2)` which swaps the integers pointed to by its two parameters. **Also write a** `main()` function that calls `swap` with appropriate parameters to illustrate how the function `swap` works. [3]

9. **Write a function** `print_triangle(n)` that prints a pattern of the following kind of shape depending on the parameter value: [6]

```
* * * * *
  * * *
    *
```

   The first row always has `2*n-1` stars and the bottom row has one star. The above illustration is the result of the call `print_triangle(3);` Note that in each row the stars are blank separated. Note how the stars are aligned. As an idea note that the first row has zero spaces at the beginning, then 5 stars separated by blanks. The second row has two spaces at the beginning then 3 stars separated by blanks. The third row has four spaces at the beginning then 1 star. You may assume that the parameter given will always be positive and it indicates the number of rows of stars to print.

10. **Write a function** `reverse(n, array1, array2)`. The parameter `n` is the number of elements in `array1`. It copies the elements of `array1` into `array2`, but in reverse order. Assume the arrays are integer arrays. [3]

11. An array has n integers. A recursive algorithm to sum the elements is: $sum(a[0]...a[n-1]) \ = \ a[0] + sum(a[1]...a[n-1])$.
    Write a recursive C function called `sum_array` using this idea. Write a main function that initializes an array to 10 integers and calls your function to compute and then print the sum. [6]

12. **Write a function** `insert(arr, n, val,pos)` which will insert an integer `val` into the integer array `arr` at the index `pos`. The parameter `n` is the number of elements currently in the array. As an example if the array contains `{ 8 4 7 1 2 }` and we call `insert(arr, 5, 9, 2)`, after the call returns the array will be `{ 8 4 9 7 1 2 }`. [6]