Fundamentals of Computing and Programming

Class Assignment 2. 20 marks, Time Limit 30 minutes

September 29, 2022

1. Write out the sequence of line numbers of statements that are executed when the following program is compiled and run: [2]

```
0  #include <stdio.h>
1  void f() {
2    printf("A\n");
3  }
4  void main() {
5   printf("B\n");
6   f();
7   f();
8   printf("C\n");
10 }
```

2. What does each of these expressions evaluate to? [4]

```
(a)  0b01100110 >> 5
(b)  0b01100110 & 0b00001111
(c)  (i=5,  ((p=pow(2,i))>16) ? p : 0)
(d)  (i=5, (i *= 10 ))
```

3. What does the following function print: [2]

```
void f( int *p1){
  int n=111;
  printf("%d",*p1);
```

```
}
void main(){
  int n=555;
  int *px = &n;
  f(px); printf("%d\n",*px);
}
```

4. What does the call stack look like (draw it) when g() is active: in particular show the value of the local variable in main(), f() and g() [2]

```
void g(int m) {
  printf("%d",m);
}
void f(int n){
  g(n-1);
}
void main(){
  int n=100;
  f(n+1)
}
```

5. What does the following program print? [2]

```
int g=100;
void f(){
  int g=200;
  g++;
  printf("1: %d\n", g);
}
void main(){
  f();
  f();
  printf("2: %d\n",g);
}
```

6. What does the above program print if we replace the definition `int g=200;` in `f()` by `static int g=200;` ? [2]

7. What does the following function print if we invoke it as f(3)? [2]

```
int f(int n){
  if (n <= 0 )
     printf("STOP ");
  else {
     printf("FLY ");
     f(n-1);
  }
}
```

8. Let $S_n = \Sigma_{i=0}^{n} 1/2^i$. A student wishes to write a C function to be invoked like `sum(n)` to compute S_n for a given parameter n. Further he wishes to write it recursively using the formulation that $S_n = S_{n-1} + 1/2^n$. **Write out the function** `sum` taking care of the base cases. (Assume, don't check, that $n \geq 0$). Use the **pow** function to compute the power of one number to another. [2]

9. Read the function below and say what `slog(10)` would print: [2]

```
void slog(int n){
   while ( n > 0 ) {
      if ( ! (n % 5 == 0) ) {
            n --;
            continue;
      }
      printf("%d ",n);
      n--;
   }
}
```