1. [6 points] Circle the correct answer among the choices given. If you circle more than one answer, you will lose the grade of the corresponding question.

(A) As the technology advances, which of the following type of algorithms can become feasible to be used?
   a. Undecidable    b. Unsolvable    c. Intractable    d. All of them

(B) ““1100” is interpreted as “-4” in which type of encoding?
   a. sign-magnitude    b. 1’s complement    c. 2’s complement
   d. (a and b)    e. (b and c)    f. (a and c)

(C) Which sorting algorithm is the worst in the best case scenario?
   a. insertion sort    b. bubble sort    c. quicksort    d. merge sort

(D) The executable is generated by:
   a. compiler    b. assembler    c. linker    d. loader    e. the programmer

(E) We have two programs A and B. In program A: function f1 calls function f2, and f2 calls f3. In program B: f1 is executed, and when it finishes f2 is executed, and when it finishes f3 is executed.
   a. Program A needs more stack    b. Program B needs more stack
   c. Both need the same stack amount    d. Function calls don’t affect stack

(F) The ALU is a:
   a. sequential circuit    b. combinational circuit
   c. we can build it with either    e. both sequential and combinational
2. (4 points) Consider the following C functions and assembly code:

```c
int fun7(int a)
{
    return a * 30;
}

int fun8(int a)
{
    return a * 34;
}

int fun9(int a)
{
    return a * 18;
}
```

Which of the functions compiles into the assembly code shown?

```
pushl %ebp
movl %esp,%ebp
movl $0(%ebp),%eax
sall $4,%eax
addl $0(%ebp),%eax
addl %eax,%eax
movl %ebp,%esp
popl %ebp
ret
```

fun8

3. (8 points) For the following logic circuit:

```
A --|   |
    |   |
    |   |
    |   |
   B   C
```

a. Draw the truth table

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>C</td>
<td>Output</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>--------</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
b. What is the function implemented with this circuit?

\[ A'B'C + AB'C + ABC \]

c. Is this function sequential or combinational? Why?

Combinational
No storage elements

4. (4 points) Assume the following MIPS code:

```
addi  R1, R0, 4
add   R2, R0,R0
addi  R4, R0, 10

Loop:   bge R2, R1, out
        bgt  R2, R4, else
        slli R2, R2, 1
        j Other

else:   addi R2, R2, 3
Other:   addi R2,R2,1
         j Loop

out:     ...
```

(i) Write down all the values that R2 will take throughout the execution of the above program. [Hint: assume the program will stop when you reach the label out].

```
0, 0, 1, 2, 3, 6, 7
```

(ii) How many times the instruction “addi R2,R2,1” will be executed?
5. (8 points) Given a sorted array A of n distinct integers, some of which may be negative, give an algorithm to find an index i such that 1 <= i <= n and A[i] = i and provided such an index exists. If there are many such indices, the algorithm can return any one of them. If no such index exists, the algorithm returns -1. Assume the array starts from element 1 (not 0). What is the best case scenario and its complexity? What is the worst case scenario and its complexity? [Note that a correct but not not optimized algorithm will get you at most 75% of the grade of this problem]

We realize that if A[i] > i then we must search elements 1 → i-1 because the array is sorted and distinct

```
search(A, start, end)
    if (start = end   AND  A[start] != start)
        return -1
    m = (start + end)/2
    if(A[m] = m )
        return m
    if(A[m] > m )
        search(A, start, m-1)
    if(A[m] < m )
        search(A, m+1, end)
```

For the above algorithm:
- Best case scenario: middle element is the solution → O(1)
- Worst case scenario: no element found → O(logn)