It is an open-book exam, but NO Coding and NO Compiling. **[Scores: 20.0p → Grade 10/100 points]**
Honour Code Signature: _____ Student Name & Number: _____
Your signature represents your promise that this quiz is solely your own work.

1. [0.5p] Consider the following four functions. Which functions shown below use the pointer correctly (select one or more)? _____

```
int *foo1(void) {
  int x = 10;
  return &x;
}
```

```
int *foo2(void) {
  int *px;
  *px = 10;
  return px;
}
```

```
int *foo3(void) {
  int *px;
  px = new int;
  *px = 10;
  return px;
}
```

```
int *foo4(void) {
  int *px;
  px = new int;
  *px = 10;
  return *px;
}
```

2. [1.0p] Fill the blank to use a function pointer `fp` to invoke `joy()` and print **"Joy: 1004"** properly.

```
#include <iostream>
int joy(int a, int b) {
  return a * b * 1004;
}
```

```
void main() {
     int _____

      fp = _____
      std::cout << "Joy:" << _____;
}
```

3. [0.5p] What does the function return when it begins with `funnies = 4`? _____

```
int funnyEars(int funnies) {
   if (funnies == 0) return 0;
   if (funnies % 2 == 0) return 3 + funnyEars(funnies - 1);
   return 2 + funnyEars(funnies - 1);
}
```

4. [0.5p] What does the `fun()` function do?

(1) x + y (2) x + x*y (3) x * y (4) x^y

```
int fun(int x, int y) {
    if (y == 0)   return 0;
    return (x + fun(x, y-1));
}
```

5. [0.5p] Consider the following recursive function `joy(x, y)`. What is the value of `joy(4, 3)`? _____

```
int joy(int x, int y) {
  if (x == 0)  return y;
  return joy(x - 1,  x + y);
}
```

6. [2.0 p] Compute the result of sum in terms of N. Then, determine the runtime complexity in Big-Oh notation.

(1)

```
int i, j, sum = 0;
for(i = 1; i <= N*N; i++)
  for(j=1; j<=i; j++)
    sum++;
```

(2)

```
int i, j, sum = 0;
for (i = 1; i <= N; i++)
    for (j = 1; j <= N; j = j * 2)
        sum++;
```

7. [1.0p] Suppose that an intermixed sequence of 10 push and 10 pop operations are performed on a stack. The pushes push the letters 0 through 9 in order; the pops print out the return value. A number in the input sequence indicates a **push**, a **dash(-)** a **pop** operation.
For example, the output **3 2 1 0 4 5 6 7 8 9** produced by **0123----4-5-6-7-8-9-** push and pops.

What operations would generate **1 0 3 4 2 7 8 6 5 9**? _____

8. [2.0p] The following code implements a digital clock. Rewrite the code in grey box to fix the problem. There are two way to fix it; one using **new**, the other without **new**.

```
#include <iostream>
#include <iomanip>
struct Clock{
  int hr, min, sec;
};

void tick(Clock *ptr) {
  ptr->sec++;
  if (ptr->sec == 60){
  // increment time by one second.
  }
}
void show(Clock *ptr)  {
  std::cout.fill('0');
  // show time in military form.
}

int main (void) {
  Clock *clock{14, 38, 56};
  for(int i = 0; i < 6; ++i) {
      tick(clock);
      show(clock);
  }
  return 0;
}
```

```
// Without using new
```

```
// With using new
```

9. [1.0p] Following is C++ like pseudo code of a function that takes a number as an argument and uses a `stack S` to do processing. What does the function `fun()` do in general?

```
void fun(int n){
    Stack S = new Stack;     // Say it creates an empty stack S
    while (n > 0) {          // while there is a number to divide
      push(&S, n%2);         // This line pushes the value of n%2 to stack S
      n = n/2;
    }
    while (!empty(&S))       // Run while Stack S is not empty
      std::cout << pop(&S) << endl; // pop an element from S and
}
```

(1) Prints binary representation of n in reverse order
(2) Prints binary representation of n
(3) Prints the values of Log n
(4) Prints the values of Log n in reverse order

10. [1.0p] Suppose you are given an implementation of a queue of integers. The operations that can be performed on the queue are:
    i. `empty(Q)` returns true if the queue is empty, false otherwise.
    ii. `delete(Q)` deletes the element at the front of the queue and returns its value.
    iii. `insert(Q, i)` inserts the integer i at the rear of the queue.

```
void fun(queue Q) {
  int i;
  if (!empty(Q)) {
    i = delete(Q);
    fun(Q);
    insert(Q, i);
  }
}
```

What operation is performed by the function `fun()` ?

(1) Leaves the queue Q unchanged
(2) Reverses the order of the elements in the queue Q
(3) Deletes the element at the front of the queue Q and inserts it at the rear keeping the other elements in the same order
(4) Empties the queue Q

11. [1.0p] Consider the following pseudocode that uses a stack.

```
declare a stack of characters
while ( there are more characters in the word to read ) {
   read a character
   push the character on the stack
}
while ( the stack is not empty ) {
   pop a character off the stack
   write the character to the screen
}
```

What is output for input "handong"?
(a) handonghandong (b) handong (c) gnodnahgnodnah (d) gnodnah

12. [1.0p] The following postfix expression with single digit operands is evaluated using an operand stack:
5 1 2 + 4 * + 3 -
(1) Evaluate the postfix expression: _____
(2) The top two elements of the stack after the first * is evaluated are: _____ _____

13. There are many way of solving recurrence equations. Sometimes our **intuition** helps us solve the problem. Let's suppose that we have the recurrence equation defined below (It came from Hanoi Tower). Compute T(N) and O(N).
$T(1) = 1$
$T(N) = 2\,T(N - 1) + 1$
This time we plug in a few cases and see how it progresses:
Hint: Let's plug in a few numbers to compute some values of in T(N).
[1.0p]
$T(1) = 1$

$T(2) =$ (answer it by number)

$T(3) =$ (answer it by number)

$T(4) =$ (answer it by number)

Did you something from the computation above? Then, you know what to do.
To get T(N), expand T(2), T(3), T(4), …, T(N) using either telescoping or unfolding.
[1.0p]

[1.0p] Your answers:

T(N-1) = _____ T(N) = _____

O(N) = _____

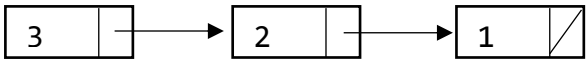
14. [1.0p] What does the following function do for a given Linked List with first node as head?

(a) Prints all nodes of linked lists
(b) Prints all nodes of linked list in reverse order
(c) Prints alternate nodes of Linked List
(d) Prints alternate nodes in reverse order

```
void joy(struct node* head) {
  if(head == nullptr) return;
  joy(head->next);
  cout << head->data << " ";
}
```

15. Assume that you have the following structures and functions available.

```
struct Node {
   int    item;
   Node*  next;
};
using pNode = Node*;
pNode push_front(pNode h, int val);
pNode pop_front(pNode h, int *val);
pNode show(pNode h);
```



(1) [1.0p] The `stacked()` function is supposed to create a link list with an int array **and** returns the first node pointer. For example, if the input array `int a[] = { 1, 2, 3 }`, the output should be as shown above and returns the first node pointer. Fix the bug in the `stacked()` in the left.

```
pNode stacked(int *a, int n) {

  for (int i = 0; i < n; i++)
    pNode h = push_front(h, a[i]);

  return h;
}
// this code has a bug.
```

```
pNode stacked(int *a, int n) {
```

(2) [1.0p] What is the time complexity of `stacked()` implemented above ? _____

(3) [2.0p] As you have seen at (1), this process is called `'push'` in the stack data structure. Implement this process as a function, `push_front(pNode h, int val)` function. Identify the `push_front()` that does **not** work properly. Some functions may have some redundant code on purpose but work OK. Select "**Blank**" if you don't know the answer since there will be a penalty for a wrong answer.

```
pNode push_front(pNode h, int val) {
  if (h == nullptr)
    return new Node {val, nullptr};
  pNode node = new Node {val, h};
  node->next = h;
  return node;
}
```

Correct/Incorrect/Blank

```
pNode push_front(pNode h, int val) {
  if (h == nullptr)
    return new Node {val, nullptr};
  return new Node {val, h};
}
```

Correct/Incorrect/Blank

```
pNode push_front(pNode h, int val) {
  pNode node = new Node {val, nullptr};
  node->next = h;
  return node;
}
```

Correct/Incorrect/Blank

```
pNode push_front(pNode h, int val) {
  return new Node {val, h};
}
```

Correct/Incorrect/Blank