18.35

THIS IS NOT AN OPEN BOOK EXAM. [TOTAL SCORES: 25 points + 2 Bonus points]
Honour Code Signature: _____ Student Name & Number: 218008015 이동한
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)? 3

```
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)(int, int);

    fp = joy
    std::cout << "Joy:" << fp(1,1) ;
}
```

1.5

3. [0.5p] What does the function return when it begins with funnies = 4? 12

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

3+2+3+2+0 = 12

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

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

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

5. [0.5p] Consider the following recursive function joy(x, y). What is the value of joy(4, 3)? 12

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

3 1
2 10
1 12
0

6. [2.0 p] Compute the result of sum in terms of N. Then, determine a tight bound of 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++)
```

N^2+1
$1+2+\cdots+N^2+1$

sum = $\frac{N^2(N^2+1)}{2} = \frac{1}{2}N^4 + \frac{1}{2}N^2$

Big-Oh = $O(N^4)$

(2)

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

$N+1$
$1, 2, 4, 8, \cdots, N$
$1+2+4+8+\cdots+N$

sum = $2^{k+1} - 1 = 2N - 1$ where $k = \lfloor \log_2 N \rfloor$

Big-Oh = $O(N)$

1.0
$N = 2^k$, $k = \log_2 N$
$k = \lfloor \log_2 N \rfloor$
$(\because j \le N)$
integer

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**? 01--23-4--567-8---9-

1.0

I love you regardless of your scores.

3.5

God is good all the time

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;
}
```

error

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

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

9. [0.5p] 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) //

(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 orde

10. [0.5p] 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() ?

(2) //

(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

1.5

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: 14
(2) The top two elements of the stack after the first * is evaluated are: 12 5

13. [3.5p] The following function which implements a binary search algorithm returns the index of the search key if the key is present in given array a[0..n-1], otherwise -1.
(1) [1.0 p] Complete the following recursive binary search algorithm.

```
int binsearch(int a[],int key, int lo, int hi) {
   if (___ hi < lo ___) return -1;
   int mid = (lo + hi)/2;

   if (key == a[mid]) return mid;
   if (key < a[mid])
      return binsearch(a, key, lo , mid-1 );

   return binsearch(a, key, mid+1 , hi );
}
int main() {
  int a[] = {97, 15, 20, 22, 24, 26, 27, 29, 30, 39, 50, 53, 57, 63, 10};
  int n = 15;
  quicksort(a, n);
  binsearch(a, 21, 0, n-1);
}
```

(2) [0.5 p] Assume that your main code is supposed to do a binary search for the key 21 and quicksort() is already implemented. Give the sequence of values (or a[mid]) in the array, that are compared with the key 21.

Answer: 29, 22, 15, 20

(3) [0.5 p] Write a line of code to replace a magic number in mian().

int n = sizeof(a) / sizeof a[0];

(4) [0.5p] What is the time complexity of binsearch() function described above? O(log n)

(5) [0.5p] What is the time complexity of main() function described above? O(1)

(6) [0.5p] Define the time complexity of the recursive binary search T(n) in recurrence equation including a constant time c. The recurrence equation may include T(n) forms.

T(n) = $c + T(\frac{n}{2}) = c + c + \cdots + T(1) = O(\log n)$

I love you regardless of your scores.

God is good all the time

3.5

14. [1.0p] 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).
$T(1) = 1$

$T(2) = 2T(1)+1$ (answer it by number) $= 3$

$T(3) = 2T(2)+1$ (answer it by number) $= 7$

$T(4) = 2T(3)+1$ (answer it by number) $= 15$

Did you something from the compuation 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.

$T(N) = 2T(N-1)+1$ $\quad (T(1)=1)$

$T(N+1) = 2T(N-2)+1$

$T(3) = 2T(T(2)+1$
$T(2) = 2T(1)+1$

$1 + 2 + 4 + \cdots + 2^{N-1}$

$= 2^N - 1$

Your answers:

T(N-1) = $2^{N-1}-1$

T(N) = $2^N - 1$

O(N) = $O(2^N)$

15. [0.5p] What does the following function do for a given Linked List with first node as head?

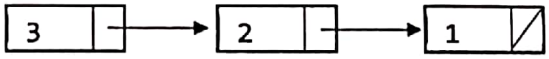
(a) Prints all nodes of linked lists
(b) Prints all nodes of linked list in reverse order
(c) Prints alternate nodes of Linked List
(d) Prints alternate nodes in reverse order

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

1.5

16. 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) {
pNode h = nullptr;
for(int i=0; i<n; i++)
    h = push_front(h,a[i]);
return h;
}
```

(2) [0.5p] What is the time complexity of stacked() implemented above ? _____

(3) [1.0p] 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 whether or not each implementation is correct, incorrect or "Blank".
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

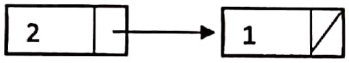
```
pNode push_front(pNode h, int val) {
  pNode node = new Node {val, nullptr};
  node->next = h;
  return node;
}
```
Correct/Incorrect/Blank

```
pNode push_front(pNode h, int val) {
  return new Node {val, h};
}
```
Correct/Incorrect/Blank

(4) [0.5p] What is the time complexity of push_front() implemented above? O(1)

2.5

(5) [1.0p] Now let's remove the first node from the stack and returns the next node which becomes at the top of the stack or new head. The item value from the removed node is also returned through the second argument **int *val**. This operation is called '**pop**' in stack. Implement in **pop_front()** function.




```
pNode pop_front(pNode h, int *val) {
```

```
pNode new_h = h->next;
delete h;
return new_h;
```

0.5

(6) [1.0p] Write **void show(pNode h)** that prints all the items from the top of the stack.

```
void show(pNode h){
    while (h != nullptr){
        cout << h->item << endl;
        h = h->next
    }
}
```

17. [1.0 p] The following code snippets are supposed to add a new node with 'val' at the end of the list. Assume that list has one node at least. Identify whether or not each implementation is correct, incorrect or "Blank". Select "Blank" if you don't know the answer since there will be a penalty for a wrong answer.

```
pNode push_back(pNode h, int val) {
    pNode x = h;
    while (x->next != NULL)
      x = x->next;
    x->next = new Node{val};
    return h;
}
```

Correct/Incorrect/Blank

0.5

```
pNode push_back(pNode h, int val) {
    pNode x = h;
    while (x != NULL) {
      if (x->next == NULL) {
        x->next = new Node{item};
        break;
      }
      x = x->next;
    }
    return h;
}
```

Correct/Incorrect/Blank

```
pNode push_back(pNode h, int val) {
    pNode x = h;
    while (x != NULL)
      x = x->next;
    x->next = new Node{val};
    return h;
}
```

Correct/Incorrect/Blank

```
pNode push_back(pNode h, int val) {
    pNode x = h;
    while (x->next != NULL) {
      x->next = new Node{val};
      x = x->next;
    }
    return h;
}
```

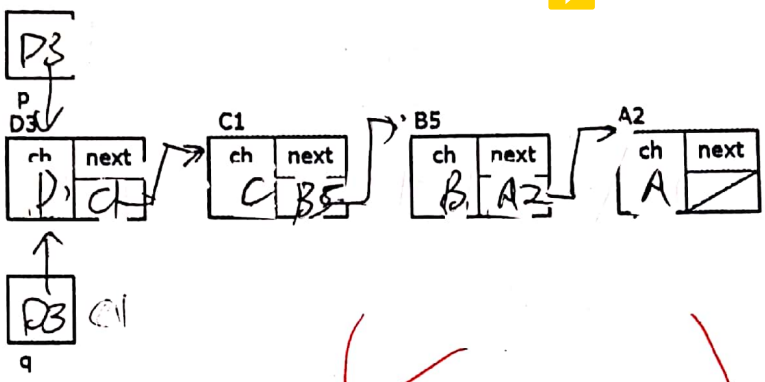
Correct/Incorrect/Blank

→ This code make all list item to val

2.0

18. [1.0p] Assuming the input `A, B, C, D` to this program, what would be the data structure **after the input after execution of the first while loop**? Complete the following figure to show a memory diagram to represent the data structures of the code. Use a mnemonic memory address to represent each node such as `A2, B5, C1, ...`, etc. Use an arrow to indicate a link between two nodes.

```
#include <iostream>
using namespace std;
struct Node {
  char  ch;
  Node* next;
};
int main( ) {
  Node* p = nullptr, *q = nullptr;
  char ch;
  while (cin.get(ch) && ch != '\n') {
    p = new Node {ch, q};
    q = p;
  }
  while (p != nullptr) {
    cout << p->ch << " ";
    p = p->next;
  }
  cout << endl;
}
```



19. [0.5p] What is the output after execution of **the second while loop**? OUTPUT D C B A

20. [1.0p] What are the values of `p`, `q` and `q->next->ch`? You may use a mnemonic memory address when necessary.

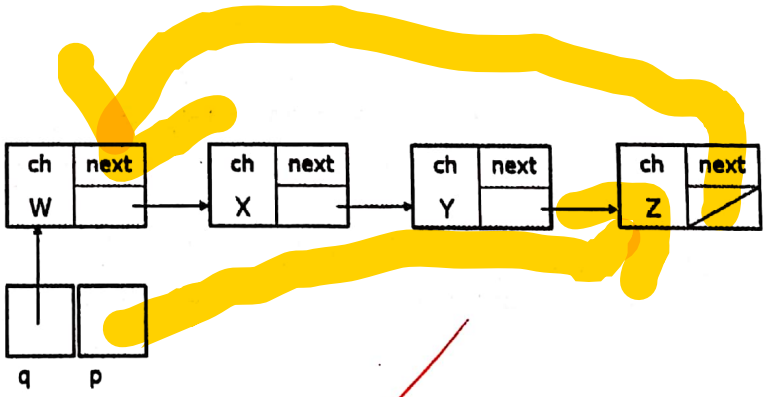
(1) Value of **p**: D3

(2) Value of **q**: D3

(3) Value of **q->next->ch**: C

21. [1.0p] Complete the following figure to show a memory diagram to represent the data structures of the code. Use an arrow to indicate a link between two nodes.

```
#include <iostream>
using namespace std;
struct Node {
  char  ch;
  Node* next;
};
int main( ) {
  Node* p = nullptr, *q = nullptr;
  ...
  // some nodes are linked as shown
  ...
  p = q;
  while (p->next != nullptr)
      p = p->next;
  p->next = q;
```



22. [0.5p] After the code block listed above is executed, what would it print if you run the following code?

```
do {
    cout << p->ch;
    p = p->next;
} while (p != nullptr);}
```

Bonus Points:

[1.0 p] Explain a few in-house programming principles discussed in class.

DRY, NMN, NSE, IIS

Do not repeat yourself. No magic number, No side effect, Implenementation and Interface separation

[1.0p] Write a function called **clear()** that deallocates all the nodes in a linked list and returns **nulltpr**. Assume that the node structure has a **next** data field to link the next node and a header of the list is passed in.

```
pNode clear(pNode h) {
    while( h != nullptr) {
        pNode nexth = h->next;
        delete h;
        h = h->next;
    }
}
```

0.75

1.75