Final Exam

C COMPUTER PROGRAMMING（I）

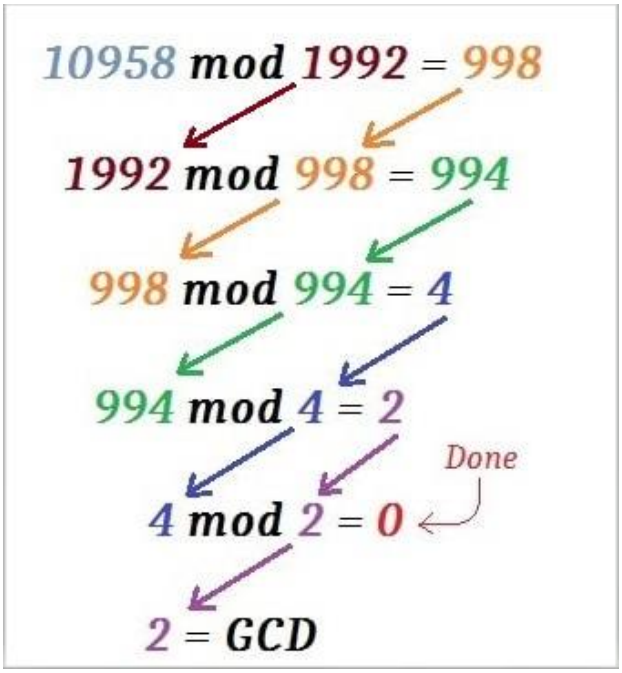
December 14, 2023

Exam rules

- Only Dev-C++ can be used for the exam.
- If your codes cannot be compiled by Dev-C++, it is considered as syntax error.
- Please save your codes as a c source file named after your student ID each problem. ex. B123456789_a.c, B123456789_b.c, B123456789_c.c …
- If your codes cannot output the desired result, your points will be deducted.
- Please check whether your codes can be compiled and output the desired result before submitting to National Sun Yat-sen Cyber University.
- No reason for late submission.
- It is forbidden to search for information during the exam.
- The cheaters will get zero point.

Please design your program to complete the following problems.

a. (20 points)
Euclidean algorithm is a technique for finding the greatest common divisor (GCD) of two numbers. GCD is the largest number that can be divided between two without leaving a remainder.



Euclidean algorithm works as follows:

1. You take two numbers. If one number is greater than another number, divide the larger number by the smaller number and take the remainder.
2. Then, replace the larger number with the smaller number, and replace the smaller number with the remainder from step 1.
3. Repeat the process: divide the new larger number by the new smaller number and take the remainder.
4. Continue this process of substitution and division until the remainder is 0.
5. When the remainder is 0, the divisor in this step is the GCD of the original two numbers.

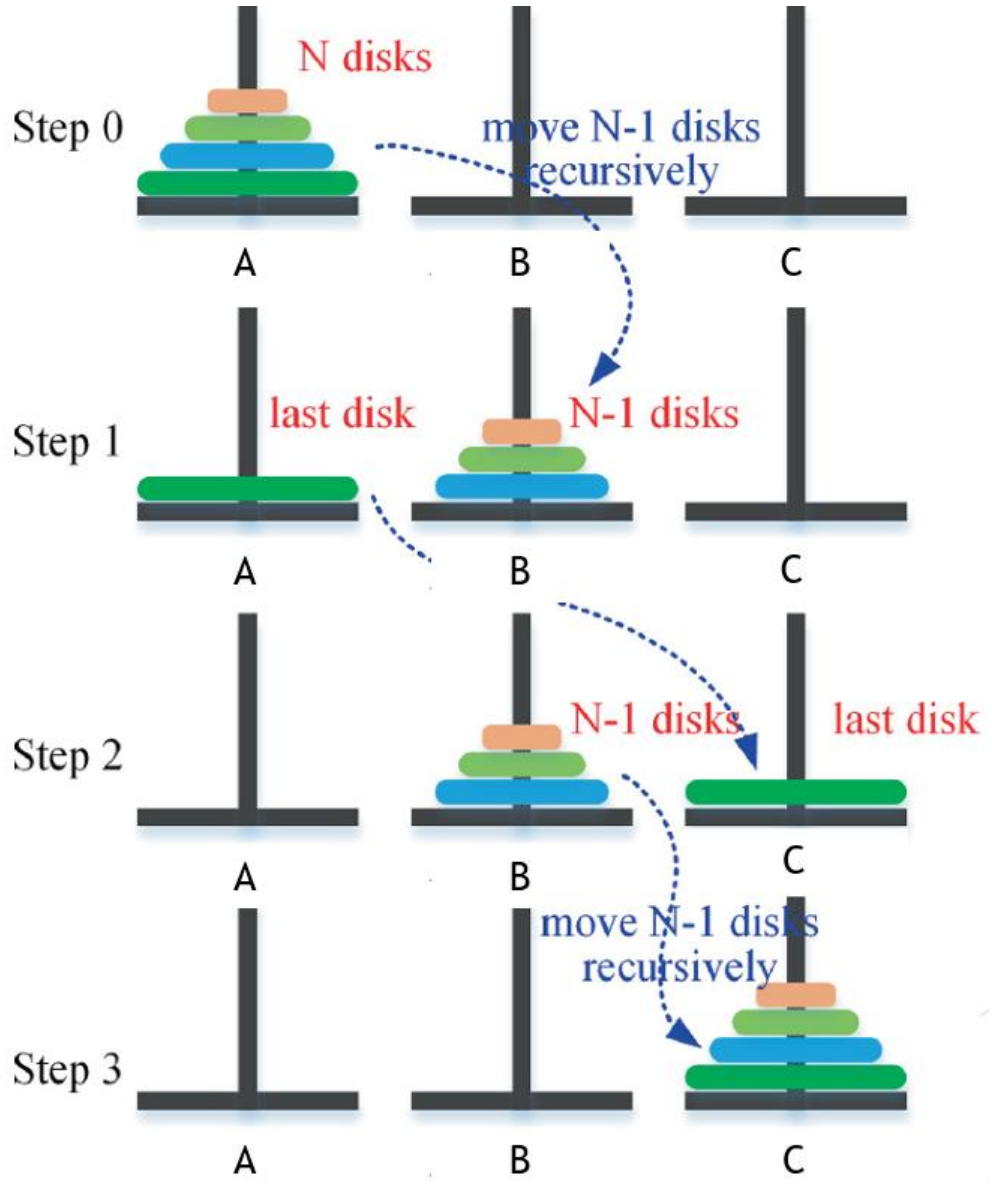
Please read the attachment "GCD.txt", where each line contains two numbers, and then implement a recursive function to calculate the GCD of each line.

Sample output:

```
The GCD of (279, 439) is 1.
The GCD of (691, 330) is 1.
The GCD of (227, 158) is 1.
The GCD of (144, 462) is 6.
The GCD of (704, 461) is 1.
The GCD of (130, 908) is 2.
The GCD of (737, 385) is 11.
The GCD of (237, 126) is 3.
The GCD of (748, 270) is 2.
The GCD of (283, 899) is 1.
The GCD of (671, 703) is 1.
The GCD of (598, 127) is 1.
The GCD of (285, 389) is 1.
The GCD of (566, 127) is 1.
The GCD of (971, 917) is 1.
```

b. (20 points)

Tower of Hanoi is a mathematical puzzle where we have three towers and n disks.



To move n disks from tower A to tower C, using tower B as the intermediary, a rudimentary algorithm would look like this:

1. Move n−1 discs from A to B
2. Move one disc from A to C
3. Move n−1 discs from B to C

Please let the user enter the number of disks, then use a recursive function to output the order and number of moves.

Note:

(1) 1 <= N <= 10

(2) Your code should be able to input and output repeatedly.

Sample output:

```
Enter the number of disks : 2
The sequence of moves involved in the Tower of Hanoi are :

 Move disk 1 from A to B
 Move disk 2 from A to C
 Move disk 1 from B to C

 The number of moves : 3

Enter the number of disks : 3
The sequence of moves involved in the Tower of Hanoi are :

 Move disk 1 from A to C
 Move disk 2 from A to B
 Move disk 1 from C to B
 Move disk 3 from A to C
 Move disk 1 from B to A
 Move disk 2 from B to C
 Move disk 1 from A to C

 The number of moves : 7
```

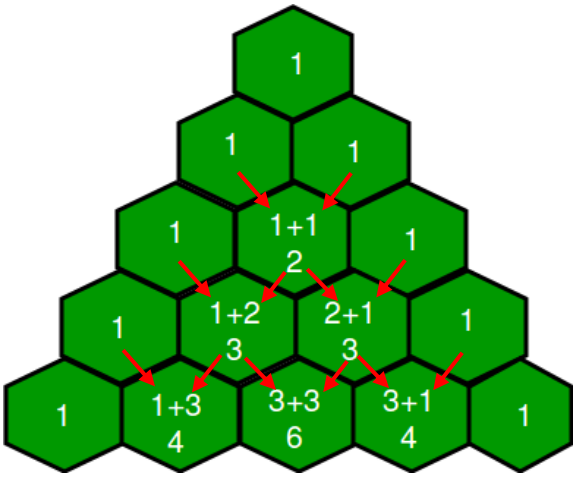
```
Enter the number of disks : 4
The sequence of moves involved in the Tower of Hanoi are :

 Move disk 1 from A to B
 Move disk 2 from A to C
 Move disk 1 from B to C
 Move disk 3 from A to B
 Move disk 1 from C to A
 Move disk 2 from C to B
 Move disk 1 from A to B
 Move disk 4 from A to C
 Move disk 1 from B to C
 Move disk 2 from B to A
 Move disk 1 from C to A
 Move disk 3 from B to C
 Move disk 1 from A to B
 Move disk 2 from A to C
 Move disk 1 from B to C

 The number of moves : 15
```

c. (20 points)

Pascal's triangle is a special triangle that is named after the French mathematician Blaise Pascal. The numbers in Pascal's triangle are placed in such a way that each number is the sum of two numbers just above the number.



Please let the user enter an integer N, then use a recursive function to output the first N levels of Pascal's triangle.

Note:

(1) 1 <= N <= 15

(2) Print each **element** using "**%5d**".

(3) Print each **whitespace character** using "**%5c**".

(4) Your code should be able to input and output repeatedly.

Sample output:

```
Please input a layer number :0
Wrong input, input again!

Please input a layer number :16
Wrong input, input again!

Please input a layer number :10
                                    1
                                1       1
                            1       2       1
                        1       3       3       1
                    1       4       6       4       1
                1       5       10      10      5       1
            1       6       15      20      15      6       1
        1       7       21      35      35      21      7       1
    1       8       28      56      70      56      28      8       1
1       9       36      84      126     126     84      36      9       1
```

d. (20 points)

Please read the attachment "Student.txt", where each line contains the student's student number, name, Chinese, English, and math scores. Then define a structure array to store the information, and then use a recursive bubble sort function to sort these elements according to different items.

Note:

(1) The maximum length of student name is **10**.
(2) Sort student ID by **ascending order**.
(3) Sort student name **from A to Z**.
(4) Sort scores by **descending order**.
(5) Implement swap operation by **pointer**.
(6) After finishes any operation, print whole array.
(7) Your code should be able to input and output repeatedly.

Sample output:

```
  ID       Name  Chi  Eng Math
1001     Olivia   77   62   71
1002       Emma   84   93   64
1003        Ava   61   63   75
1004     Sophia   56   89   70
1005   Isabella   68   85   89
1006  Charlotte   93   67   76
1007     Amelia   79   83   75
1008        Mia   77   78   75
1009     Harper   69   57   55
1010     Evelyn   99   63   58
1011       Liam   90   88   59
1012       Noah   64   58   54
1013     Oliver   83   93   61
1014     Elijah   70   92   61
1015    William   86   52   56
1016      James   55   52   96
1017   Benjamin   59   60   81
1018      Lucas   65   53   70
1019      Henry   72   83   96
1020  Alexander   76   65   91
```

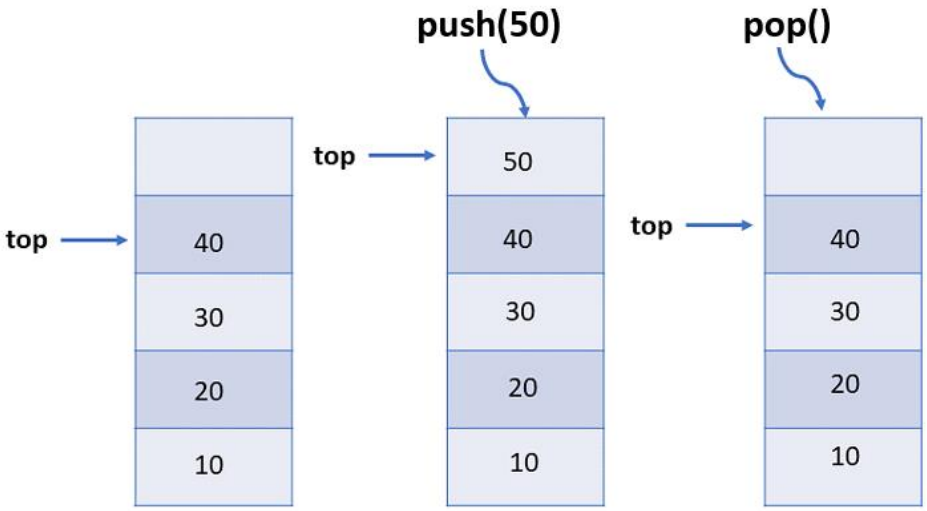
```
Choose sorting method
1. ID 2. Name 3.Chinese 4.English 5.Math 6.Exit: 2
   ID       Name  Chi  Eng Math
 1020  Alexander   76   65   91
 1007     Amelia   79   83   75
 1003        Ava   61   63   75
 1017   Benjamin   59   60   81
 1006  Charlotte   93   67   76
 1014     Elijah   70   92   61
 1002       Emma   84   93   64
 1010     Evelyn   99   63   58
 1009     Harper   69   57   55
 1019      Henry   72   83   96
 1005   Isabella   68   85   89
 1016      James   55   52   96
 1011       Liam   90   88   59
 1018      Lucas   65   53   70
 1008        Mia   77   78   75
 1012       Noah   64   58   54
 1013     Oliver   83   93   61
 1001     Olivia   77   62   71
 1004     Sophia   56   89   70
 1015    William   86   52   56
```

```
Choose sorting method
1. ID 2. Name 3.Chinese 4.English 5.Math 6.Exit: 3
   ID       Name  Chi  Eng Math
 1010     Evelyn   99   63   58
 1006  Charlotte   93   67   76
 1011       Liam   90   88   59
 1015    William   86   52   56
 1002       Emma   84   93   64
 1013     Oliver   83   93   61
 1007     Amelia   79   83   75
 1008        Mia   77   78   75
 1001     Olivia   77   62   71
 1020  Alexander   76   65   91
 1019      Henry   72   83   96
 1014     Elijah   70   92   61
 1009     Harper   69   57   55
 1005   Isabella   68   85   89
 1018      Lucas   65   53   70
 1012       Noah   64   58   54
 1003        Ava   61   63   75
 1017   Benjamin   59   60   81
 1004     Sophia   56   89   70
 1016      James   55   52   96
```

e. (20 points)

A stack is a linear data structure in which nodes can be inserted and removed from only one side of the linked list.

In a stack, we always use a pointer called top to track the last node present in the linked list.



Please read the attached file "Stack.txt" to implement a linked list containing 100 different integer nodes, and then provide a function menu to perform the stack operations:

1. **Push** : Insert new node with an integer at the top of the stack.
   *You can only add new node using **malloc()**
2. **Pop** : Remove the top node from the stack and return the value.
   *You should check whether stack is empty, then release memory using **free()**
3. **Reverse** : Reverse the order of nodes in the stack by **push** and **pop**.
4. **Search** : Check if a node containing the target value in the stack.
   Returns the node position if it exists, otherwise outputs not found.
   *The first node with the target value is the target node.

Note:

(1) You should perform any operation by **function call**.
(2) After finishes any operation, print whole stack.
(3) Print each element using "**%7d**", newline each 5 elements.
(4) Your code should be able to input and output repeatedly.

Sample output:

```
 99999->  99102->  98051->  97128->  96201->
 95154->  94132->  93107->  92009->  91226->
 90133->  89156->  88123->  87092->  86210->
 85215->  84121->  83201->  82174->  81247->
 80198->  79121->  78112->  77014->  76199->
 75201->  74114->  73157->  72183->  71131->
 70092->  69105->  68178->  67219->  66095->
 65140->  64207->  63107->  62168->  61315->
 60170->  59282->  58276->  57213->  56128->
 55109->  54137->  53115->  52065->  51100->
 50196->  49209->  48047->  47125->  46012->
 45301->  44047->  43314->  42237->  41302->
 40163->  39139->  38105->  37051->  36120->
 35208->  34179->  33138->  32225->  31021->
 30154->  29101->  28006->  27007->  26115->
 25108->  24123->  23174->  22211->  21089->
 20149->  19126->  18155->  17039->  16104->
 15239->  14126->  13001->  12084->  11120->
 10245->   9132->   8010->   7087->   6029->
  4961->   3782->   2395->   1203->    394->
```

```
Please select the action
1. push  2. pop 3. reverse 4. search: 1
enter an integer : 15

    15->  99999->  99102->  98051->  97128->
 96201->  95154->  94132->  93107->  92009->
 91226->  90133->  89156->  88123->  87092->
 86210->  85215->  84121->  83201->  82174->
 81247->  80198->  79121->  78112->  77014->
 76199->  75201->  74114->  73157->  72183->
 71131->  70092->  69105->  68178->  67219->
 66095->  65140->  64207->  63107->  62168->
 61315->  60170->  59282->  58276->  57213->
 56128->  55109->  54137->  53115->  52065->
 51100->  50196->  49209->  48047->  47125->
 46012->  45301->  44047->  43314->  42237->
 41302->  40163->  39139->  38105->  37051->
 36120->  35208->  34179->  33138->  32225->
 31021->  30154->  29101->  28006->  27007->
 26115->  25108->  24123->  23174->  22211->
 21089->  20149->  19126->  18155->  17039->
 16104->  15239->  14126->  13001->  12084->
 11120->  10245->   9132->   8010->   7087->
  6029->   4961->   3782->   2395->   1203->
   394->
```

```
Please select the action
1. push  2. pop 3. reverse 4. search: 2
The value at the top is 15.


  99999->  99102->  98051->  97128->  96201->
  95154->  94132->  93107->  92009->  91226->
  90133->  89156->  88123->  87092->  86210->
  85215->  84121->  83201->  82174->  81247->
  80198->  79121->  78112->  77014->  76199->
  75201->  74114->  73157->  72183->  71131->
  70092->  69105->  68178->  67219->  66095->
  65140->  64207->  63107->  62168->  61315->
  60170->  59282->  58276->  57213->  56128->
  55109->  54137->  53115->  52065->  51100->
  50196->  49209->  48047->  47125->  46012->
  45301->  44047->  43314->  42237->  41302->
  40163->  39139->  38105->  37051->  36120->
  35208->  34179->  33138->  32225->  31021->
  30154->  29101->  28006->  27007->  26115->
  25108->  24123->  23174->  22211->  21089->
  20149->  19126->  18155->  17039->  16104->
  15239->  14126->  13001->  12084->  11120->
  10245->   9132->   8010->   7087->   6029->
   4961->   3782->   2395->   1203->    394->
```

```
Please select the action
1. push  2. pop 3. reverse 4. search: 3

    394->   1203->   2395->   3782->   4961->
   6029->   7087->   8010->   9132->  10245->
  11120->  12084->  13001->  14126->  15239->
  16104->  17039->  18155->  19126->  20149->
  21089->  22211->  23174->  24123->  25108->
  26115->  27007->  28006->  29101->  30154->
  31021->  32225->  33138->  34179->  35208->
  36120->  37051->  38105->  39139->  40163->
  41302->  42237->  43314->  44047->  45301->
  46012->  47125->  48047->  49209->  50196->
  51100->  52065->  53115->  54137->  55109->
  56128->  57213->  58276->  59282->  60170->
  61315->  62168->  63107->  64207->  65140->
  66095->  67219->  68178->  69105->  70092->
  71131->  72183->  73157->  74114->  75201->
  76199->  77014->  78112->  79121->  80198->
  81247->  82174->  83201->  84121->  85215->
  86210->  87092->  88123->  89156->  90133->
  91226->  92009->  93107->  94132->  95154->
  96201->  97128->  98051->  99102->  99999->
```

```
Please select the action
1. push  2. pop 3. reverse 4. search: 4
enter an integer : 393
Stack did not exist a node contains 393.
```

```
Please select the action
1. push  2. pop 3. reverse 4. search: 4
enter an integer : 1203
No.2 node contains 1203.
```