中原大學 109 學年度 ☑上學期 考試解答 ☑期中

科目名稱：資料結構　　開課班級：資工二甲/資工二乙　　油印份數：72/68

I. Single-Choice Problems (50%) 每題 3 分，共 20 題，答錯一題倒扣 1 分，超出 50 分以 50 分計

1	2	3	4	5	6	7	8	9	10
A	B	D	D	A	C	B	C	A	C
11	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
B	C	C	A	B	C	D	D	A	D

II. Simple-Answering Problems (30%)每一空格 3 分，共 11 空格，無倒扣，超出 30 分以 30 分計

(1) aList.getLength()　　(2) remove　　(3) aList.getLength()+1　　(4) aList.getLength()

(5) aList.getLength()-i+2 (**Reason**: i=1, aList.getLength()+1; i=2, aList.getLength(); ...)

(6) There are exactly 8 strings in total: 1a0 1b0 11a 11b a00 b00 0a1 0b1

(7) No. (**Reason**: ~~1~~100a111~~0~~ ➔ ~~1~~00a111 ➔ ~~0~~0a11~~1~~ ➔ ~~0~~a1~~1~~ ➔ a1➔ no match!)

(8) Example answers (of exactly two rules) just for reference, where ε means an empty string:（請勿部份給分！）

<S> = 01 | 0<X>1	<S> = 01 | 0<X>1	<S> = 0<X>1	<S> = 0<X>1
<X> = 0 | 1 | 0<X> | 1<X>	<X> = 0 | 1 | <X>0 | <X>1	<X> = ε | 0 | 1 | 0<X> | 1<X>	<X> = ε | 0 | 1 | <X>0 | <X>1

(9) push　　(10) pop　　(11) false

III. Advanced Problems (20%)每一空格 3 分，共 7 空格，無倒扣，超出 20 分以 20 分計

Example answers just for reference:（關鍵的程式碼不可欠缺，請勿部份給分！）

(1) Inheritance: clas baseC {int w; public: int x; void y() {...} }; class derivedC: public baseC {int z;};

(2) overloading: class anyC {void sameF(int pInt) {…} void sameF(float pFloat) {…}};

(3) overriding: clas baseC {public: void sameF() { … } }; class derivedC: public baseC { void sameF() {…}};

(4) 4 9 8 3 1 + / 9 7 - + - * 6 +

(5) + [4 9 8 4] ➔ / [4 9 2] ➔ - [4 9 2 2] ➔ + [4 9 4] ➔ - [4 5] ➔ * [20] ➔ + [26]

(6) Do the same way from right to left on the prefix expression: + * 4 - 9 + / 8 + 3 1 – 9 7 6

- [6 2] ➔ + [6 2 4] ➔ / [6 2 2] ➔ + [6 4] ➔ - [6 5] ➔ * [6 20] ➔ + [26]

(7)（關鍵的程式碼不可欠缺，請勿部份給分！）

```
#include <iostream>
#include <vector>
using namespace std;

#define LEN 3             // triplet length

int   triplet[LEN];       // the current triplet
int   intSum = 15;        // the required sum
vector<int> intA = {1, 5, 4, 9, 6};
void find_triplets(int, int);

int main()
{     find_triplets(0, 0);
      return 1;
}
```

```
void find_triplets(int p, int n)
{     if (n < LEN)
            for (int i = p; i < intA.size(); ++i )
            {     triplet[n] = intA[i]; // add the next one
                  find_triplets(i+1, n+1);
            }
      else // Base case: produce a triplet of length LEN
      {     int   tsum = 0;           // temporary sum

            for (int j = 0; j < LEN; ++j )
                  tsum += triplet[j];
            if (tsum == intSum)       // qualified
            {     for (int j = 0; j < LEN; ++j )
                        cout << triplet[j] << " ";
                  cout << endl;
            }
      }
}
```