CITY UNIVERSITY OF HONG KONG

Course code & title : CS2311 Computer Programming

Session : Semester B 2020/21

Time allowed : Two hours

This paper has 15 pages (including this cover page).

---

1. This paper consists of four questions.
2. Answer ALL questions.

---

*This is an **open-book** examination.*

Academic Honesty

I pledge that the answers in this exam are my own and that I will not seek or obtain an unfair advantage in producing these answers. Specifically,

- *I will not plagiarize (copy without citation) from any source;*
- *I will not communicate or attempt to communicate with any other person during the exam; neither will I give or attempt to give assistance to another student taking the exam; and*
- *I will use only approved devices (e.g., calculators) and/or approved device models.*
- *I understand that any act of academic dishonesty can lead to disciplinary action.*

I pledge to follow the Rules on Academic Honesty and understand that violations may led to severe penalties.

Student ID: _____ Student Signature: _____

CS Departmental Hotline (phone, whatsapp, wechat): +852 6375 3293

Q1 (30%)	Q2 (20%)	Q3 (30%)	Q4 (20%)	Total (100%)

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

Source: https://simple.wikipedia.org/wiki/ASCII

Question 1 (30%)

Each of the following programs contains bugs to prevent them to meet the aim of the program and produce the expected output (if given to you). Modify the programs to fix the bugs by **inserting your code changes in the box provided.** Note that each program may contain **one or more bugs.** Also **explain** your logics of the changed code so that the change can fix the bug(s)**.**

The aim of this program is to compute the sum of values kept in the array `s[]`. The expected output is 6.

Example Program	**Code Change (Example)**

`#include <iostream>`
`using namespace std;`
 	
`int main()`
`{`
`  int s[3] = { 1, 2, 3};`
`  int sum = 0;`
`  for (i = 1; i <= 3; i++)`
`  sum = sum + s[i];`
`  cout << sum << endl;`
`  return 0;`
`}`
	`for (int i = 0; i < 3; i++)`

My Explanation

The array contains three elements, and in C++, it starts storing elements at the index position of 0. So, we should only go through s[0], [1], and [2] instead of s[1], s[2], s[3] to read the values of the array.

(a) The aim of this program is to convert the lowercase and uppercase letters in the cstring `"CS2311 Computer Programming"` to the uppercase and lower letters, respectively. The expected output is: `cs2311 cOMPUTER pROGRAMMING`

Program for 1(a)	**Code Change**

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

int main()
{
     char s[] = "CS2311 Computer Programming";
     int diff = 'A' - 'a';
     char z = 'z', Z = 'Z';

     int i = 0;
     while (s[i]) {
          if (s[i] <= '@') {}
          else if (s[i] <= z)
               s[i] = s[i] - diff;
          else if (s[i] <= Z)
               s[i] = s[i] + diff;
          i++;
     }
     cout << s << endl;
}
```

My Explanation

(b) The aim of this program is to print all negative numbers in the integer array `scores[]`. The expected output of the program is: `-42`

Program for 1(b)	**Code Change**

```
#include <iostream>
using namespace std;
#define SIZE 5

void printNeg(int* x, int size) {
     int i = 0;
     while (i < size) {
          if (*x < 0)
               cout << *x << endl;
          x++;
          size++;
     }
}

int main() {
     int size = SIZE;
     int scores[size] = { 75,  64, -42, 2};
     int* ptr = scores;
     printNeg(size, ptr);
}
```

My Explanation

(c) The aim of this program is to print the smallest numbers inputted. For instance, if a user inputs the value of 4 followed by 4 integers (40, 20, 20, 50), the expected output is: `20`.

Program for 1(c)	**Code Change**

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

int main(){
	int n,result,i,x;

	int a[10] = {0};
	n = 0;
	do {
		cin >> n;
	} while (n <= 0 && n >= 10);

	for (i = 0; i < n, n < 10 ; i++) {
		cin >> a[i];
	}
	result = a[0];

	while (i < n) {
		if (result < a[i])
			result = a[i];
		i++;

	cout << result;
}
```

My Explanation

(d) The aim of this program is to ask for inputting the three sides of two triangles, constructs the two corresponding triangle objects in C++ using the class Triangle, compare the areas of these two triangle objects, and print whether or not the areas of these two objects are the same.

Program for 1(d)	**Code Change**

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

class Triangle {
    int sideA, sideB, sideC;
public:

    int area();
};

int Triangle::area() {
    return sideA * sideA * sideA;
};



int main() {
    int a1 = 0, b1 =0, c1 =0, a2=0, b2=0, c2=0;
    cin >> a1 >> b1 >> c1; // for triangle one
    cin >> a2 >> b1 >> c2; // for triangle two
    Triangle t1(a1, b1, c1);
    Triangle t2(a2, b2, c2);
    if (t1.sideA == t2.sideA &&
        t1.sideB == t2.sideB  &&
```

```
        t1.sideC == t2.sideC)
        cout << "same size" << endl;
    else
        cout << "different size" << endl;
}
```

My Explanation

Question 2. (20%)

Write a program to accept a positive odd integer `N` (e.g., 3, 5, 7, 9) and print the shape 'X' such that both the width and the height of the shape are `N`. The input values are underlined in the following four examples. **Explain your code.**

```
Input: 3
* *
 *
* *
```

```
Input: 5
*   *
 * *
  *
 * *
*   *
```

```
Input: 7
*     *
 *   *
  * *
   *
  * *
 *   *
*     *
```

```
Input: 9
*       *
 *     *
  *   *
   * *
    *
   * *
  *   *
 *     *
*       *
```

My C++ program is as follows:

Explanation of the logics of the code

Question 3 (30%)

A maze is a 5x5 grid, in which each cell contains an integer. This integer indicates the possible direction(s) that may move out from that cell. The rows from top to bottom are rows 0, 1, 2, 3, and 4, respectively; and the columns from left to right are columns 0, 1, 2, 3, and 4, respectively. We refer to a cell as (x,y) where x = 0...4 and y=0…4. In Example 1, the cell (4,4) contains 0, and the cell (4, 2) contains 6. **The maze always starts at the cell (0, 0).**

The rule of the maze is as follows:

1. If a cell contains a **positive** number (e.g., 2), then we may move to the cell below it. E.g., in Example 2, we can move from (2, 3) to (3, 3).
2. If a cell contains a **negative** number (e.g., -1), then we may move to the cell above it. E.g., in Example 2, we can move from (3, 2) to (2, 2).
3. If a cell contains an **even** number (e.g., 2), then we may move to the cell on its right. E.g., in Example 2, we can move from (0, 0) to (0, 1).
4. If a cell contains an **odd** number (e.g., 9), then we may move to the cell on its left. E.g., in Example 2, move from (3, 3) to (3, 2).
5. If the cell contains 0 (zero), then it is a destination.
6. At any time, we cannot move out of the maze.

Write a program to satisfy all of the following requirements and **explain** the strategy of your code to be developed to compute the solution.

- The program accepts a sequence of 25 integers as its input to populate the maze from top to bottom and from left to right (i.e., from (0,0), (0, 1), …, (4, 3), (4,4)).
- Output "`There is a solution.`" if it is possible to have a path in the maze starting from (0,0) to reach any cell containing 0. If there is no feasible path, output "`No solution.`"
    - **Hint that your program needs to not to find out any exact path.**

The input values are underlined in the following three examples.

Example 1

```
1 2 3 4 5
1 2 3 4 5
1 2 3 4 5
1 2 3 4 5
2 4 6 8 0
There is a solution.
```

Example 2

```
 2  1 -3  4 -2
 3  4  2 -8  1
 5 -4  7  2  1
-1 -1 -1  9 -1
 0  0  0  0  0
There is a solution.
```

Example 3

```
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
No solution.
```

For your reference only: Two paths that can be found for Example 1 and Example 2 are shown below. The diagram on the right shows all the cells that can be reached from (0.0).

Example 1

```
1 2 3 4 5
1 2 3 4 5
1 2 3 4 5
1 2 3 4 5
2 4 6 8 0
```

Example 2

```
 2  1 -3  4 -2
 3  4  2 -8  1
 5 -4  7  2  1
-1 -1 -1  9 -1
 0  0  0  0  0
```

Reachable cells in Example 2

```
 2  1     4 -2
 3  4  2 -8  1
 5 -4  7  2  1
-1 -1 -1  9 -1
          0
```

My strategy used to find whether there are any feasible paths are as follows:

My C++ program is:

Question 4 (20%)

The file **dictionary.txt** is a word dictionary, which contains 1000 lines, and each line contains exactly one English word. Write a program to meet the following requirements:

- The program accepts one word from Console.
- It determines whether the input word is defined in the word list kept in the dictionary.
- It handles file management properly.
- It keeps the words in the dictionary in an array named dict[].
- It outputs the checking result to Console.
- It **does not use** any string library function (e.g., strcmp, strnmp).
- It **only uses pointers** to update dict or read from dict. (i.e., do not use syntax like dict[i] to refer to the element kept in dict[i].)
    - Note: If your program uses such array syntaxes, marks will be deducted.
- You may assume that each word contains no more than 20 characters.

dictonary.txt

```
hello
world
cs2311
computer
programming
...
```

Example 1

hello
hello is in the dictionary.

Example 2

cs2311
cs2311 is in the dictionary.

Example 3

Hello
Hello is not in the dictionary.

Example 4

Exam
Exam is not in the dictionary.

```
My C++ program is:
```

*** END ***