Mock Exam

Song Liu (song.liu@bristol.ac.uk)
GA 18, Fry Building,
Microsoft Teams (search "song liu").

Instructions

- Please finish the following questions within 1 hour and 30 mins.
- Try to complete the exam without checking lecture slides/lab code.
- **Part A**: 5 small questions, 8 points each.
- **Part B**: 2 big questions, 30 points each.
- **Hint**: Big questions may not be as hard as some of the small questions. If you know the answers to big questions, do them first.

Part A, 1.1

(3 marks)

Which one of the following statements about von Neumann architecture is wrong?

1. von Neumann architecture includes CPU, memory and IO devices.
2. Computational tasks are performed by CPU.
3. CPU manages data stored in the memory.
4. A touch screen is not an IO device.
5. A hard rive is an IO device.

Answer: 4

Part A, 1.2

(3 marks)

Which component in von Neumann architecture is responsible for executing your program?

Answer: CPU

Part A, 1.3

(2 marks)

Name **one** low level programming language and **one** high level programming language.

Answer: Machine Code and C/C++

Part A, 2.1

(2 marks)

Explain the main differences between Procedure Programming (PP) and Object Oriented Programming (OOP).

Answer: PP **divides your program into subtasks** and you write procedures for each subtask. OOP **divides your program into objects** which contains methods and fields.

Part A, 2.2

(3 marks) Which one of the following statement is wrong?

1. High level programming language requires compilation before execution.
2. C is a low level programming languages.
3. C++ is both a PP and an OOP language.
4. A valid C program is a valid C++ program.
5. In OOP, objects can inherit methods and fields from its parent class.

Answer: 2

Part A, 2.3

(3 marks)
In OOP, what are the two components that can be contained in an object?

Answer: Methods and Fields (or Operations and Data).

Part A, 3.1

(2 marks)
What do `malloc` and `free` functions do? When should they be used?
Answer: `malloc` allocates **heap memory** for your program while `free` releases the **heap memory** you just allocated.

They should be used when you want to dynamically allocate memory for your variables.

Part A, 3.2

(6 marks)

Read the following code:

```
#include <stdio.h>
void goo(int a){
    printf("goo is being called!\n");
    printf("%d", a);
}
void koo(int a){
    printf("koo is being called!\n");
    goo(a)
}
void foo(){
    int a = 0;
    koo(2);
    printf("koo has been called!\n");
}
void main(){
    foo();
}
```

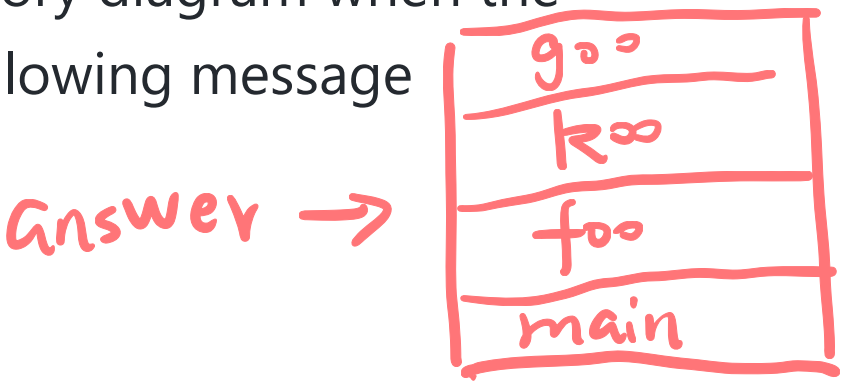
Part A, 3.2

1. (2 marks) Write down the display output after the above code is executed.

   Answer:

```
koo is being called!
goo is being called!
2
koo has been called!
```

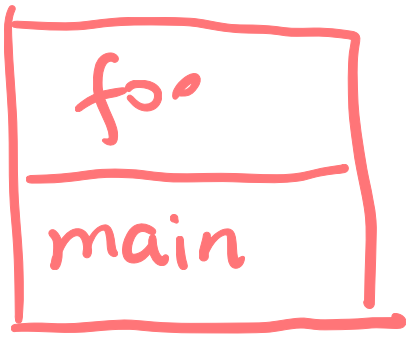
2. (3 marks) Draw the stack memory diagram when the program is printing out the following message `goo is being called!`.



Answer →

3. (3 marks) Draw the stack memory diagram when the program is printing out the following message `koo has been called!`.

Answer →



Part A, 4

Read the following C++ code:

```
#include <stdio.h>
class shop_item{
    //write your code below
};

int main(){
    shop_item coke;
    //write your code below
}
```

Part A, 4

1. (2 marks) Add three **private** fields in `shop_item` class: `name` stores a string, `quantity` stores a integer, `price` stores a decimal number.

   Answer:

   ```
   class shop_item{
       char* name;
       int quantity;
       float price;
   };
   ```

   By default, fields are private.

Part A, 4

2. (3 marks) Add **public** methods in `shop_item` class, so that you can set values to the fields you have just defined.
    - Note, `quantity` and `price` must be positive.

      Answer:

```
class shop_item{
    ...
public:
    void set_name(char* n){name = n;}
    void set_quantity(int q){
        if(quantity > 0){
            quantity = q;
        }
    }
    void set_price(int p){
        if(price >0){
            price = p;
        }
    }
};
```

Part A, 4

3. (3 marks) Add code in `main` function, so that `coke`'s name is set to `"coca cola"`, `quantity` is set to 100 and `price` is set to `1.2f`.

   Answer:

```
int main(){
    shop_item coke;
    coke.set_name("coca cola");
    coke.set_quantity(100);
    coke.set_price(1.2f);
}
```

Part A, 5

A twin prime pair are two prime numbers that are only 2 apart. For example, (3, 5), (5, 7), (11, 13) are all twin primes.

Part A, 5

```
n <- 100
last_prime <- 2

for(i in 2:n){
  is_prime <- TRUE
  for(j in 2:(i-1)){
    if(_____){
      is_prime <- FALSE
    }
  }

  if(is_prime){
    if(______){
        # print out twin prime pair.
        print(paste(last_prime,',',i))
    }
    last_prime <- i
  }
}
```

Part A, 5

1. (3 marks) Fill out the blanks, so that the above code identifies all twin primes smaller than 100.
   Answer: first blank
   `i%%j == 0`
   second blank
   `i - last_prime == 2`

2. (3 marks) If the above code takes 2s to run, how long it will take if `n` is set to 1000. Why?
   Answer: 200 seconds. The computational complexity is quadratic with respect to n, so the computation time is 2* (1000/100)^2 = 200.

Part A, 5

3. (2 marks) Modify/Add one line of code, so the above above program will run significantly faster.
   Answer: You can change
   `for(i in 2:n){` to `for(i in c(2, seq(3,n,2))){`
   so you are looping over only odd numbers.
   or You can add
   `break`, after `is_prime <- FALSE`. Either is fine.

Part B, 1

1. (30 marks) Write a complete C program which produces the following output:

```
*
**
***
*
**
***
*
**
***
```

   You must use `for` loops.

Part B, 1

Answer:

```
#include <stdio.h>
void main(){
    for(int i = 0; i < 9; i=i+1){
        for(int j = 0; j < i%3+1; j=j+1){
            printf("*");
        }
        printf("\n"); // do not forget this.
    }
}
```

Hint: You need nested two for loops: The outer for loop prints out multiple rows, and the inner for loop controls the display within each row.

Even if you cannot write answers in full, please write as much as possible. If the first for loop is written correctly and the the code is compilable, you will get 15 points.

Part B, 2

2. (30 marks) Complete the following R program, so that it produces a matrix D whose i, j-th entry $D_{i,j}$ is the Euclidean distance between the i-th row of A and j-row of B. Euclidean distance between two K dimensional vectors are defined as $\mathrm{dist}(\boldsymbol{a}, \boldsymbol{b}) = \sqrt{\sum_{k=1}^{K}(a_k - b_k)^2}$.

```
A <- matrix(rnorm(100*2), nrow = 100)
B <- matrix(rnorm(100*2), nrow = 100)
D <- matrix(0, nrow = 100, ncol = 100)
# Your code starts here
```

Answer:

```
for(i in 1:100){
    for(j in 1:100){
        D[i,j] <- sqrt(sum((A[i,] - B[j,])^2))
    }
}
```

or any code that computes `D`.

If you cannot write the full code, please write as much as possible.

In the exam, the program may/may not be vectorizable. You do not need to vectorize if you feel that the program cannot be vectorized.