1. C Fundamentals. **(10 marks)**

(a) Declare a macro symbolic constant `SPEED` with a single-precision floating point value 3.25×10^{-26}. **(2 marks)**

```
#define SPEED 3.25e-26f
```

(b) A C program contains the following declarations: **(2 marks)**

```
int i, j;
long ix;
short s;
float x;
```

What is the resulting **data type** of the expression

```
(int) ix / s - 2.5 * 'Z' + x * i / j ?
```

double

(c) Consider the following C program: **(2 marks)**

```
#include <stdio.h>

int func(int a, int b)
{
    return a * --b;
}

int main(void)
{
    int i = 5;
    int j = 2 * func(1+2, i+1);
    printf("%d %d", i, j);
    return 0;
}
```

What is the output of the program?

5 30

(d) Re-write `func(int a, int b)` in program in (c) into a function-like macro `FUNC(A, B)`, such that when the call to `func(1+2, i+1)` in the program is replaced with `FUNC(1+2, i+1)`, the outputs will remain the same. **(2 marks)**

```
#define FUNC(A,B) ((A)*((B)-1))
```

(e) What is the output of the following C program? **(2 marks)**

```
#include <stdio.h>

int main(void)
{
    for (int i = 0; i < 10; i++)
        if (i % 3)
            printf("%d ", i);
    return 0;
}
```

```
1 2 4 5 7 8
```

2. Arrays, Strings and Pointers. **(20 marks)**

(a) Using only one C statement, declare an array which can hold 100 integers with initial values 1, 2, 3 and 4 for the first four elements, and 0 for the remaining elements. Name this array `iarray`. **(2 marks)**

```
int iarray[100] = {1, 2, 3, 4};
```

(b) Declare a C string symbolic constant named `STRING` using macro with value `"Hello, world"` **(2 marks)**

```
#define STRING "Hello, world"
```

(c) Consider the following C program: **(2 marks)**

```
#include <stdio.h>

#define S1 "ABCD"
#define S2 "1234"

int main(void)
{
    char *str = S1 S2;
    printf("%s", str);
    return 0;
}
```

Will the program compile? Briefly explain your answer.

There are no syntax errors in the program, hence it will compile.

(d) Consider the following C code snippet:

```
char carray[] = "ABCD\0WXYZ";
```

i. What is the value of the expression `sizeof(carray)`? **(2 marks)**

10

ii. What is the value of the expression `strlen(carray)`? **(2 marks)**

4

(e) Given the following array and pointer declarations:

```
short sarray[] = {1,2,4,8,16};
short *sp = sarray;
short **spp = &sp;
```

Suppose that a `short` occupies 2 bytes. The address of `sarray` is 100, while `sp` and `spp` are at addresses 200 and 204, respectively (all addresses are expressed in decimal form).

i. What is the numeric value of the expression `sarray`? **(2 marks)**

100

ii. What is the numeric value of the expression `sp + 2`? **(2 marks)**

100 + 2(2) = 104

iii. What is the numeric value of the expression `*(sp + 2)`? **(2 marks)**

sp[2] = sarray[2] = 4

iv. What is the numeric value of the expression `*sp + 2`? **(2 marks)**

sp[0] + 2 = sarray[0] + 2 = 1 + 2 = 3

v. What is the numeric value of the expression `*spp + 3`? **(2 marks)**

100 + 3(2) = 106

3. User-Defined Types and Linked Lists. **(12 marks)**

(a) Consider the following C code snippet:

```
enum maker { toyota, honda, mercedes = 10, audi };

union info {
    int year;
    char age;
};

struct car {
    enum maker maker;
    char model[10];
    union info info;
};
```

i. What is the value of the symbolic constant honda? **(2 marks)**

1

ii. Using only one C statement, declare a variable named c1 which is of type struct car, and initialize the members maker and model to audi and "a4", respectively. **(2 marks)**

```
struct car c1 = {audi, "a4"};
```

iii. Suppose that for the variable c1 declared in the (ii), the following assignment statement is given: **(2 marks)**

```
c1.info.year = 2017;
```

What will be the value of c1.info.age? Briefly explain your answer.

c1.info.age will have invalid/garbage/rubbish value because it shares storage with c1.info.year

iv. Assuming that an int occupies 32 bits, what is the size (in bytes) of the variable c1 declared in (ii)? Provide a brief explanation of your answer. **(2 marks)**

The size of c1 is equal to

sizeof(enum maker) + 10*sizeof(char) +
max(sizeof(char, sizeof(int))).

Now, sizeof(enum maker) = sizeof(int), hence the size of c1 is 4 + 10 + max(1, 4) = 18 bytes.

(b) Consider a singly-linked list which contains a list of integers. A node in this list is defined as follows: **(4 marks)**

```
struct node {
    int data;
    struct node *next;
};
```

Suppose that `head` points to the head of the list.

Write the necessary C code to print all the elements of the list, beginning at the head.

```
struct node *p = head;
while(p != NULL) {
    printf("%d", p->data);
    p = p->next;
}
```

4. Storage Classes and Dynamic Memory. **(13 marks)**

(a) Briefly describe what memory leak is and its effect on the memory usage of a program. **(3 marks)**

Memory leak is the loss of pointer to allocated memory. Because of this, the program will not be able to free the memory, causing its memory usage to keep on increasing.

(b) Consider the following C code snippet: **(2 marks)**

```
int *mem = (int *)calloc(10, sizeof(int));
for(int i=0; i<10; i++) {
    /* Do something with mem[i] */
}

realloc(mem, 20*sizeof(int));
for(int i=0; i<20; i++) {
    /* Do something with mem[i] */
}
```

Briefly describe one problem (if there is any) with the above code, and how you should fix it. Assume that the call to `realloc()` is successful.

After the call to `realloc()`, `mem` is pointing to an invalid memory location. The code can be fixed by assigning the return value of `realloc()` to `mem`.

(c) Given the following C program:

```
#include <stdio.h>

int func (int x)
{
    static int y;

    if (x == 0)
        y += 5;
    else if (x == 1)
        y += 10;
    else
        y++;

    return y;
}
```

```
int z;

int main (void)
{
    int w;

    func(1);
    func(2);
    w = func(z);
    printf("%d", w);

    return 0;
}
```

i. What is the initial value of y? **(2 marks)**

0

ii. In which memory segment is z stored? **(2 marks)**

Data segment

iii. What is the storage class of w? **(2 marks)**

Auto or automatic

iv. What is the output of the program? **(2 marks)**

16

5. C File I/O. **(5 marks)**

Suppose you are given a text file named `input.txt` which contains two decimal numbers per row:

```
1 6
12 9
3 15
30 17
```

Complete the following C program (by filling in the blanks beside comments L1, L2, L3, L4 and L5) which is supposed to add the numbers in every row and output the numbers, together with the sum to a text file named `output.txt`. The contents of `output.txt` should look like this:

```
1 6 7
12 9 21
3 15 18
30 17 47
```

```
#include <stdio.h>

int main(void)
{
    FILE *in, *out;

    /* Open input.txt for reading */
    in = _______________________________;  /* L1 */
    if (in == NULL) {
        printf("Failed to open input file."\n);
        return 0;
    }
    /* Open output.txt for writing (overwrite existing contents) */
    out = _______________________________; /* L2 */
    if (out == NULL) {
        printf("Failed to open output file."\n);
        return 0;
    }

    while( !feof(____________ /* L3 */) ) {
        int a, b;
        /* Use a and b to store decimal numbers in a row */

        int ret = fscanf(in, "%d %d", &a, &b);
        if(____________ /* L4 */) {
            int sum = a + b;
            fprintf(_________________________________ /* L5 */);
        }
    }
    fclose(in);
    fclose(out);
    return 0;
}
```

Answers:

```
1.  fopen("input.txt", "r")
2.  fopen("output.txt", "w")
3.  in
4.  ret == 2 (or ret > 1 or ret >= 2)
5.  fprintf(out, "%d %d %d\n", a, b, sum)
```

6. Process Management and Socket Programming. **(10 marks)**

(a) Name the family of system calls that does not return control back to the calling point on termination. **(1 mark)**

`exec()` system call

(b) What are the two types of sockets supported by the `socket` system call? **(2 marks)**

SOCK_DGRAM and SOCK_STREAM

(c) How many times will the following C progrm print `Hi`? **(2 marks)**

```
#include<stdio.h>
#include <sys/types.h>
#include <unistd.h>

int main()
{
    fork() && fork();
    fork();
    fork() && fork();
    printf("Hi\n");
}
```

18

(d) You are given the following C program:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/wait.h>

int gvar = 20;

int main(void)
{
    int lvar = 40;
    pid_t pid;

    if ((pid = fork()) < 0) {
        printf("fork error\n");
    } else if (pid == 0) {
        gvar++;
        lvar++;
    } else {
        wait(NULL);
    }

    printf("%ld %d %d\n", (long)getpid(), gvar, lvar);
    exit(0);
}
```

Answer the following TWO (2) questions:

i. Explain how the `fork()` function executes and how and when the variables `gvar` and `lvar` change. **(3 marks)**

After a successful fork() call, two copies of the original code executes. Both processes not only run the same program, but they resume execution as though both had called the system call fork().

Parent and child will have their own copies of variables. Variable changes in one process will not affect the variables in the other process. The values of gvar and lvar remains 20 and 40 in the parent process, while those in child changes to 21 and 41 respectively.

ii. Assume that the `fork()` is successful and that the parent process ID is 16232 while the child process ID is 16233. What is the output of the program? **(2 marks)**

```
16233 21 41
16232 20 40
```

7. C++ Classes. **(24 marks)**

(a) Define a class `rational` with the following members:

- two private integer data members: `numerator` and `denominator`.
- a `public` constructor that takes in two integers and assigns to the members `numerator` and `denominator` using an initializer list.
- a public member function `getFloat` with no arguments that returns a floating point number which is the quotient of `numerator` and `denominator`.

(5 marks)

```
class rational {
private:
    int numerator, denominator;
public:
    rational(int n, int d) :  numerator(n),
        denominator(d)
    float getFloat() {
        return float(numerator)/denominator;
    }
};
```

(b) Consider the following C++ class declaration. **(5 marks)**

```
namespace nsA {
    class classA {
    public:
        virtual void disp() const = 0;
        virtual int get (void) = 0;
        classA (int x) : a(x) {}
    protected:
        int a;
    };
}
```

Declare a class `classB` that extends `classA` but in a different namespace named `nsB`. Class `classB` should:

- preserve the access specifier of the inheritable members of `classA`.
- not be abstract.
- have an inline default constructor that initializes the member variable a to 100.

You do not need to show function implementations, just give the function prototype declarations.

```
namespace nsB {
    using namespace nsA;
    class classB:public classA {
    public:
        void disp() const
        int get (void)
        classB():classA(100) {}
    };
}
```

(c) What is the output of the following C++ program? **(3 marks)**

```
#include<iostream>

void Execute(int &x, int y = 50)
{
    int TEMP = x + y;
    x += TEMP;
    if (y != 200)
        std::cout<<TEMP<<" "<<x<<" "<<y<<std::endl;
}

int main()
{
    int A = 10, B = 20;
    std::cout<<A<<" "<<B<<std::endl;
    Execute(A, B);
    std::cout<<A<<" "<<B<<std::endl;
    return 0;
}
```

```
10 20
30 40 20
40 20
```

(d) What is the output of the following C++ program? **(4 marks)**

```
#include<iostream>

using namespace std;

class base {
```

```
public:
    base() { cout<<"BCon"<<endl; }
    ~base() { cout<<"BDest"<<endl; }
};

class derived: public base {
public:
    derived() { cout<<"DCon"<<endl; }
    ~derived() { cout<<"DDest"<<endl; }
};

int main()
{
    derived obj;
    return 0;
}
```

```
BCon
DCon
DDest
BDest
```

(e) List one difference between a `friend` function of a class and a `member` function of a class. **(2 marks)**

(Any one)

- Friend function is a non-member function that has access to private and protected members of a class. It is not in the scope of the class in which it is declared. While, a member function is in scope of the class in which it is declared.
- A friend function cannot be called using object of the class. While a member function is called using object of the class.
- A friend function can be declared in private, public or protected scope of the class without any effect. While, a member function can be declared in private, public or protected scope of the class but have scope accordingly.

(f) Given the following definition of a class `Distance`, write a definition of the `friend` function `sumDistance()` that finds sum of two `Distance` objects. (One Feet = 12 inches) **(5 marks)**

```
class Distance //English Distance class
{
private:
    int feet;
    float inches;
public:
    Distance(int f, float i) : feet(f), inches(i) {}
    Distance() {}
    void showdist() //display distance
    { std::cout << feet << "\'-" << inches << '\"'; }

    friend Distance sumDistance(Distance d1, Distance d2);
};
```

```
Distance sumDistance(Distance d1, Distance d2)
{
    Distance d;
    d.feet = d1.feet + d2.feet;
    d.inches = d1.inches + d2.inches;
    if(d.inches >= 12){
        d.inches = d.inches - 12;
        d.feet = d.feet + 1;
    }
    return d;
}
```

8. Dynamic Memory in C++ **(6 marks)**

(a) Give two reasons why `new` and `delete` are the preferred methods for managing dynamic memory in C++. **(4 marks)**

- new and delete call the constructors and destructors of objects, whereas malloc, calloc, realloc and free do not.
- new is typesafe and throws an error if it fails, and can be overloaded.

(b) Give a C++ statement which uses the `delete` operator to deallocate memory allocated by the following statement ? **(2 marks)**

```
employee *elist = new employee[10];
```

```
delete[] elist;
```

9. Templates and Containers. **(15 marks)**

(a) In C++, Standard Template Library (STL) has container classes to implement two types of lists. What are the names of these container classes? **(2 marks)**

Forward list and list

(b) What will be the output of the following C++ program? **(4 marks)**

```
#include <iostream>
#include <list>
#include <iterator>

using namespace std;

// Print the elements in a list
void showlist(list <int> l)
{
    list <int> :: iterator it;
    for(it = l.begin(); it != l.end(); ++it)
        cout << *it<<" ";
    cout <<"\n";
}

int main()
{
    list <int> list1;
    for (int i = 1; i < 10; ++i) {
        list1.push_back(i);
    }
    list1.pop_front();
    list1.reverse();
    showlist(list1);
    return 0;
}
```

9 8 7 6 5 4 3 2

(c) In C++, what does the Standard Template Library (STL) define? **(3 marks)**

Containers, Algorithms, Iterators

(d) Give three advantages of using the generic `vector` class over a C-style `array`. **(3 marks)**

Possible answers:

- There is a single definition of the vector container, but it can be used to define many different kinds (data types) of vectors.
- Vectors are similar to dynamic arrays with the ability to resize automatically when an element is inserted or deleted.
- Their storage is handled automatically by the container.
- Vectors also have safety features that make them easier to use than arrays, automated bounds checking and memory management.
- Vector elements are placed in contiguous storage so that they can be accessed and traversed using iterators.

(e) Write a generic function to return the minimum of two parameters/arguments. **(3 marks)**

```
template<typename T> // or template<class T>
T min(T a, T b) {
    return a < b ?  a :  b;
}
```

10. File I/O in C++. **(5 marks)**

(a) Write a C++ code that will declare and open a binary file `picture.gif` for input. **(2 marks)**

```
ifstream ifs;
ifs.open ("picture.gif", ios::binary);
```

(b) In C++, what is the command to clear an output stream buffer? **(1 mark)**

ostream::flush

(c) Write a statement in C++ that will read an entire line from keyboard until a `newline` character is entered and store it in a `std::string` variable `str`. **(2 marks)**

```
std::getline(std::cin, str); OR
std::getline(std::cin, str, '\n');
```

* * * * * * * * * * * * * * *