TIC1001 — Introduction to Computing and Programming
National University of Singapore

Tutorial 06

Question 1

What is the output when the following lines of code are executed?

In [1]:
```
char * capitalize(char *s)
{
    for (int i = 0; s[i]; i++)
        if ('a' <= s[i] and s[i] <= 'z')
            s[i] -= 32;
    return s;
}

string capitalize(string s)
{
    for (int i = 0; i < s.size(); i++)
        if ('a' <= s[i] and s[i] <= 'z')
            s[i] -= 32;
    return s;
}

int main(void)
{
    char cstring[20] = "Hello World!";
    char *new_cstr;
    string stdstring = "Hello World!", new_stdstr;

    new_cstr = capitalize(cstring);
    new_stdstr = capitalize(stdstring);

    printf("C-string: %s\n", cstring);
    printf("New C-string: %s\n", new_cstr);
    cout << "std::string: " << stdstring << endl;
    cout << "New std::string: " << new_stdstr << endl;
}
```

1.

Declare char array `cstring[20]`
Declare c str `new_cstr`
Declare string `stdstring`
Declare string `new_stdstr`

In [ ]:
```
char cstring[20] = "Hello World!";
char *new_cstr;
string stdstring = "Hello World!", new_stdstr;
```

1. Call function `capitalize(char * s)`, where `* s` is `cstring[20]`:

In [ ]:
```
cstring[20] = {'H','e','l','l','o',' ','W','o','r','l','d','!','\0'}
```

In [ ]:
```
char * capitalize(cstring[20])
{
    for (int i = 0; s[i]; i++)
        if ('a' <= s[i] and s[i] <= 'z')
            s[i] -= 32;
    return s;
}
```

Loop from `cstring[0]` until `cstring[i]` is no longer true

When `cstring[i]` is within 'a' and 'z' (ASCII), minus 32 to convert to upper case.

After the loop, return array and pass value of `capitalize(cstring)` to `new_cstr`:

In [ ]:
```
cstring[20] = {'H','E','L','L','O',' ','W','O','R','L','D','!','\0'}
```

In [ ]:
```
new_cstr = {'H','E','L','L','O',' ','W','O','R','L','D','!','\0'}
```

1. Call function `capitalize(string s)`, where `s` is `stdstring`:

In [ ]:
```
stdstring = {'H','e','l','l','o',' ','W','o','r','l','d','!','\0'}
```

In [ ]:
```
string capitalize(string s)
{
    for (int i = 0; i < s.size(); i++)
        if ('a' <= s[i] and s[i] <= 'z')
            s[i] -= 32;
    return s;
}
```

`s.size() = stdstring.size()` = [13]

Loop from `stdstring[0]` until `stdstring[12]`.

When `stdstring[i]` is within 'a' and 'z' (ASCII), minus 32 to convert to upper case.

After the loop, return array and pass value of `capitalize(stdstring)` to `new_stdstr`:

In [ ]:
```
std_string = {'H','E','L','L','O',' ','W','O','R','L','D','!','\0'}
```

Capitalized in the scope of `capitalize(string s)` only

In [ ]:
```
new_stdstring = {'H','E','L','L','O',' ','W','O','R','L','D','!','\0'}
```

1. Output `cstring`, `new_cstr`, `stdstring`, `new_stdstr`

In [ ]:
```
printf("C-string: %s\n", cstring);
printf("New C-string: %s\n", new_cstr);
cout << "std::string: " << stdstring << endl;
cout << "New std::string: " << new_stdstr << endl;
```

i. At this point `cstring` is:

```
In [ ]: cstring[20] = {'H','E','L','L','O',' ','W','O','R','L','D','!','\0'}
```

Output 0: C-string: HELLO WORLD!

ii. At this point `new_cstr` is:

```
In [ ]: new_cstr = {'H','E','L','L','O',' ','W','O','R','L','D','!','\0'}
```

Output 1: New C-string: HELLO WORLD!

iii. At this point `stdstring` is:

```
In [ ]: stdstring = {'H','e','l','l','o',' ','W','o','r','l','d','!','\0'}
```

Note: a copy of `stdstring` was substituted into `capitalize(string s)` , hence only the duplicated version of `stdstring` was being capitalized, while the original string array stays the same.

Output 2: std::string: Hello World!

iii. At this point `new_stdstr` is:

```
In [ ]: new_stdstring = {'H','E','L','L','O',' ','W','O','R','L','D','!','\0'}
```

Note: The capitalized `stdstring` was being returned to `new_stdstr` inside the scope of `capitalize(string s)`

Output 3: New std::string: HELLO WORLD!

Final Outputs:

C-string: HELLO WORLD!
New C-string: HELLO WORLD!
std::string: Hello World!
New std::string: HELLO WORLD!

When the signature of the second function is changed to

```
In [ ]: string capitalize(string &s)
        {
            for (int i = 0; i < s.size(); i++)
                if ('a' <= s[i] and s[i] <= 'z')
                    s[i] -= 32;
            return s;
        }
```

```
In [ ]: string &capitalize(string &s)
        {
            for (int i = 0; i < s.size(); i++)
                if ('a' <= s[i] and s[i] <= 'z')
                    s[i] -= 32;
            return s;
        }
```

`stdstring` will be passed in by address, hence updating the array after capitalization.

Question 3

Provide an implementation for the function insert using only the member functions .push_back, .pop_back and .size.

```
In [ ]: #include <iostream>
        #include <vector>

        using namespace std;

        void insert(vector<int> &v, int pos, int val)
        {

            //Loop through vector to look for index (pos)
            for (int i = 0; i < v.size(); i++)
            {
                //pos found
                if (i == pos)
                {
                    //Add one space to the back of vector
                    v.push_back(val);
                    //Move back one position, looping from the last index
                    for (int j = v.size(); j > pos ; j--)
                    {
                        v[j] = v[j-1];
                    }
                    //Replace value at pos index to val
                    v[i] = val;
                }
            }
        }

        int main (void)
        {
            vector<int> my_vector = {0, 1, 2, 3, 4, 5};

            insert(my_vector, 2, 10); // inserts into index 2 erase(my_vector, 5);
            for (int i = 0; i < my_vector.size(); i++)
                cout << my_vector[i] << " ";
        }
```

Provide an implementation for the function erase using only the member functions .push_back, .pop_back and .size.

In [ ]:

```cpp
#include <iostream>
#include <vector>

using namespace std;

void erase(vector<int> &v, int pos)
{

    //Loop through vector to look for index (pos)
    for (int i = 0; i < v.size(); i++)
    {
        //pos found
        if (i == pos)
        {
            //Run through from pos index
            for (int j = pos; j < v.size() ; j++)
            {
                //Starting from pos, replace each index with the next digit
                v[j] = v[j+1];
            }
            //Remove last digit
            v.pop_back();
        }
    }
}

int main (void)
{
    vector<int> my_vector = {0, 1, 2, 3, 4, 5};

    erase(my_vector, 2); // removes element at index 5

    for (int i = 0; i < my_vector.size(); i++)
        cout << my_vector[i] << " ";
}
```

Question 4

The function void mutate takes in a vector of int, and sets each element to the sum of its neighbours, with the ends wrapping around. For example, the vector 0, 1, 2, 3, 4, 5 will mutate to 1+5, 0+2, 1+3, 2+4, 3+5, 4+0 . Provide an implementation for the function void mutate .

In [ ]:

```cpp
#include <iostream>
#include <vector>

using namespace std;

void mutate(vector<int> &vec)
{
    //Create a duplicate, as the loop will change the vector's value
    vector<int> vec_copy = vec;

    for (int i = 0; i < vec.size(); i++)
    {
        //For first index, create wrap around
        if (i == 0)
        {
            //Sum of last index + next index
            vec[i] = vec_copy.back() + vec_copy[i+1];
        }
        //For last index, create wrap around
        else if (i == vec.size()-1)
        {
            //Sum of previous index + first index
            vec[i] = vec_copy[i - 1] + vec_copy.front();
        }
        //The rest
        else
        {
            vec[i] = vec_copy[i-1] + vec_copy[i+1];
        }
    }
}

int main (void)
{
    vector<int> my_vector = {0, 1, 2, 3, 4, 5};

    mutate(my_vector);

    for (int i = 0; i < my_vector.size(); i++)
        cout << my_vector[i] << " ";
}
```