# Programming in Modern C++: Assignment Week 1

Total Marks : 25

Partha Pratim Das
Department of Computer Science and Engineering
Indian Institute of Technology Kharagpur, Kharagpur – 721302
partha.p.das@gmail.com

January 26, 2025

## Question 1

Consider the following program. *[MCQ, Marks 2]*

```
#include <iostream>
#include <cstring>
#include <stack>

using namespace std;

int main(){
    char str[10] = "COMPUTER";
    stack<char> s1, s2;
    int i;
    for(i = 0; i < strlen(str)/2; i++)
        s1.push(str[i]);
    for(; i < strlen(str); i++)
        s2.push(str[i]);

    while (!s1.empty()) {
        s2.push(s1.top()); s1.pop();
    }
    while (!s2.empty()) {
        cout << s2.top(); s2.pop();
    }
    return 0;
}
```

What will be the output?

a) `COMPUTER`

b) `RETUPMOC`

c) `UTERCOMP`

d) `COMPRETU`

**Answer**: d)
**Explanation:**
The stack `s1` stores {P, M, O, C} and the stack `s2` stores {R, E, T, U}. Then the elements of `s1` also pushed into `s2` as {C, O, M, P, R, E, T, U}. Thus, when we finally pop and print the elements from `s2`, the output would be `COMPRETU`.

## Question 2

Which of the following is NOT a container adapter? *[MCQ, Mark 1]*

a) `stack`

b) `queue`

c) `deque`

d) `priority_queue`

**Answer**: c)
**Explanation:**
All of them are sequence containers.
However, `stack`, `queue` and `prority_queue` are the sequence containers adapted with specific protocols of access like LIFO, FIFO, Priority and known as container adaptors.

## Question 3

Consider the following code segment. *[MCQ, Marks 2]*

```
#include <iostream>
#include <algorithm>
using namespace std;
int main() {
    char data[] = {'a', 'b', 'c', 'd', 'e'};
    char key = 'd';
    if(binary_search(____________________))    //LINE-1
        cout << "found";
    else
        cout << "not found";
    return 0;
}
```

Identify the appropriate option/s to fill in the blank `LINE-1` such that output becomes `found`.

a) `&data[0], &data[5], key`

b) data, data+5, key

c) data, key, data+5

d) `&data[0], &key, &data[5]`

**Answer**: a), b)
**Explanation:**
`binary_search(.)` function takes 3 parameters. The first two parameters are staring and ending address of the array, and the third parameter is the key to search. Hence, the correct option is a) and b).

## Question 4

Consider the following code segment. *[MCQ, Marks 2]*

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

void modify(int *arr){
    rotate(arr, arr + 3, arr + 5);
    rotate(arr, arr + 2, arr + 4);
}

int main() {
    int iarr[5];
    for(int i = 0; i < 5; i++)
        *(iarr + i) = i + 1;

    modify(iarr);
    for (int i = 0; i < 5; ++i)
        cout << *(iarr + i) << " ";
    return 0;
}
```

What will be the output?

a) `1 2 3 4 5`

b) `1 2 4 5 3`

c) `4 5 1 2 3`

d) `2 3 4 5 1`

**Answer**: b)
**Explanation:**
`rotate(first, middle, last)` rotates the order of the elements in the range $[first, last]$, in such a way that the element pointed by middle becomes the new first element.
`rotate(arr, arr + 3, arr + 5)` makes the order as `4 5 1 2 3`.
`rotate(arr, arr + 2, arr + 4)` makes the order as `1 2 4 5 3`.

## Question 5

Consider the following code segment. *[MCQ, Marks 2]*

```
#include <iostream>
#include <vector>
int main() {
    std::vector<int> cVec(3, -1);
    for(int i = 0; i < 3; i++)
        cVec[i] = (i + 1) * 10;
    cVec.resize(3);
    cVec.resize(3, 110);
    for(int i = 0; i < 3; i++)
        cVec.push_back((i + 1) * 20);
    for(int i = 0; i < cVec.size(); i++)
        std::cout << cVec[i] << " ";
    return 0;
}
```

What will be the output?

a) `10 20 30 20 40 60`

b) `-1 -1 -1 10 20 30 20 40 60`

c) `-1 -1 -1 10 20 30 0 0 0 20 40 60`

d) `10 20 30`

**Answer**: a)
**Explanation:**
Vectors are similar to dynamic arrays having the ability to resize itself automatically when an element is inserted or deleted, with their storage being handled automatically by the container. The statements and the states of the vector are as follows:

`std::vector<int> cVec(3, -1);` => `[-1 -1 -1]`,

```
for(int i = 0; i < 3; i++)}
    cVec[i] = (i + 1) * 10;  => [10 20 30]
```

`cVec.resize(3);` => `[10 20 30]`
Also, `cVec.resize(3, 110)` => `[10,20,30]`

```
for(int i = 0; i < 3; i++)}
    cVec.push_back((i + 1) * 20); => [10 20 30 20 40 60]}
```

## Question 6

Consider the following code segment. *[MSQ, Marks 2]*

```
#include <iostream>
#include <algorithm>
using namespace std;
int main() {
    int iarr[] = {10, 20, 50, 40, 10, 50};
    rotate(&iarr[0], &iarr[2], &iarr[6]);
    ____________________________;    //LINE-1
    for(int i = 0; i < 4; ++i)
        cout << iarr[i] << " ";
    return 0;
}
```

Fill in the blank at `LINE-1` with appropriate option/s such that the output is: `40 10 10 20`

a) `remove(&iarr[0], &iarr[6], 50)`

b) `remove(&iarr[0], &iarr[5], 50)`

c) `remove(iarr, iarr + 6, 50)`

d) `remove(iarr, iarr + 6, iarr[5])`

**Answer**: a), c)
**Explanation:**
After execution of `rotate(&iarr[0], &iarr[2], &iarr[6])`, the contents of array becomes `50 40 10 50 10 20` .
For the desired output, the value `50` has to be removed, which can be done using the statement:
`remove(&iarr[0], &iarr[6], 50)`
or `remove(iarr, iarr + 6, 50)`

## Question 7

Consider the following code segment. *[MCQ, Marks 2]*

```
#include <iostream>
#include <algorithm>
using namespace std;
int main() {
    int idata[] = { 1, 2, 3, 4, 5 };
    int n = sizeof(idata) / sizeof(*idata);
    for (int i = 0; i < n/2; i++) {
        int temp = idata[i];
        replace(idata, idata + 5, temp, *(idata + n - i - 1));
        replace(idata + i + 1, idata + 5, idata[n - i - 1], temp);
    }
    for (int i = 0; i < 5; ++i)
        cout << idata[i] << " ";
    return 0;
}
```

What will be the output?

a) `3 2 1 2 3`

b) `1 2 3 2 1`

c) `5 4 3 4 5`

d) `5 4 3 2 1`

**Answer**: d)
**Explanation:**
In the first iteration,
the statement: `replace(idata, idata + 5, temp, *(idata + n - i - 1));`
replaces the left-most element (left-most element is saved as `temp`) with the right-most element,
and the statement: `replace(idata + i + 1, idata + 5, idata[n - i - 1], temp);`
replaces the right-most element with `temp`.
This will be continued till the middle of the array. Thus, it reverses the array.

## Question 8

Consider the following code segment. *[MSQ, Marks 2]*

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

int main(void) {
    string str1 = "Modern ";
    string str2 = "C++";
    ____________________;    //LINE-1
    cout << str1;
    return 0;
}
```

Choose the appropriate option to fill in the blank at `LINE-1`, such that the output of the code would be: `Modern C++`.

a) `str1 += str2`

b) `strcat(str1, str2)`

c) `str1.append(str2)`

d) `str1.insert(str2)`

**Answer**: a), c)
**Explanation:**
In C++, `operator+=` and `append(·)` append the strings. Please note that `strcat(·)` is a C function, and required inclusion of `cstring`. The function `insert(·)` is used to insert a string in another string at a given position.

## Question 9

Consider the following code segment. *[MCQ, Marks 2]*

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

int main() {
    int iArr[] = {40, 50, 10, 30, 20};
    sort (iArr, iArr + 4);
    for (int i = 0; i < 5; i++)
    cout << *(iArr + i) << " ";
    return 0;
}
```

What will be the output?

a) `10 20 30 40 50`

b) `10 30 40 50 20`

c) `50 40 30 20 10`

d) `50 40 30 10 20`

**Answer**: b)
**Explanation:**
Since in the `sort(·)` function, the end is set after the 4th element, it sorts from the first position to fourth position, leaving the last element of the array. Thus, the output is `10 30 40 50 20`.

# Programming Questions

## Question 1

Consider the program below.

- Fill in the blank at `LINE-1` to include appropriate header file to utilize `sqrt(·)` function.
- Fill in the blank at `LINE-2` to compute the length between two points `p1` and `p2` as $\sqrt{(p1.y - p2.y)^2 + (p1.x - p2.x)^2}$.

The program must satisfy the given test cases. *Marks: 3*

```
#include <iostream>
__________________     //LINE-1
using namespace std;
struct point{
    int x, y;
};

double get_len(point p1, point p2){
    return ______________________;    //LINE-2
}

int main() {
    int x1, y1, x2, y2;
    cin >> x1 >> y1 >> x2 >> y2;
    point p1, p2;
    p1.x = x1;
    p1.y = y1;
    p2.x = x2;
    p2.y = y2;
    cout << get_len(p1, p2);
    return 0;
}
```

### Public 1

```
Input: 10 10 20 20
Output: 14.1421
```

### Public 2

```
Input: 10 20 40 20
Output: 30
```

### Private

```
Input: 20 40 60 10
Output: 50
```

**Answer:**

```
LINE-1:  #include <cmath>
LINE-2:  sqrt((p1.y - p2.y) * (p1.y - p2.y) + (p1.x - p2.x) * (p1.x - p2.x))
```

**Explanation**:

The C library `math.h` can be included in C++ program as

```
#include <cmath>
```

At `LINE-2`, the formula to compute the distance between two points can be implemented as:

```
sqrt((p1.y - p2.y) * (p1.y - p2.y) + (p1.x - p2.x) * (p1.x - p2.x))
```

## Question 2

Consider the following program.

- Fill in the blank at `LINE-1` with the appropriate header of `max_str( )`.
- Fill in the blank at `LINE-2` with the appropriate statements such that the string array can be sorted in descending order.

The program must satisfy the sample input and output. *Marks: 3*

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

_____________________________ {    //LINE-1
    _________________;            //LINE-2
}

int main() {
    std::string words[5], word;
    for(int i = 0; i < 5; i++){
        cin >> word;
        words[i] = word;
    }
    sort(words, words + 5, max_str);
    for (int i = 0; i < 5; i++)
        cout << words[i] << " ";
    return 0;
}
```

### Public 1

```
Input: Lion Tiger Bear Hippo Bull
Output: Tiger Lion Hippo Bull Bear
```

### Public 2

```
Input: Wale Fox Deer Frog Crab
Output: Wale Frog Fox Deer Crab
```

### Private

```
Input: Dog Cat Bat Pig Cow
Output: Pig Dog Cow Cat Bat
```

**Answer:**

```
LINE-1:  bool max_str (string s1, string s2)
LINE-2:  return (s1 > s2)
```

**Explanation**:

At `LINE-1`, define function header as:

```
bool max_str (string s1, string s2)
```

At `LINE-2`, define the return statement for sorting in descending order as:

```
return (s1 > s2);
```

## Question 3

Consider the following program.

- Fill in the blanks at `LINE-1` to add each string to the stack.
- Fill in the blanks at `LINE-2` to print the element at the top of the stack.
- Fill in the blanks at `LINE-3` to remove the element at the top of the stack.

The program must satisfy the sample input and output. *Marks: 3*

```
#include <iostream>
#include <stack>
#include <vector>

void printReverseOrder(std::vector<std::string> words){
    std::stack<std::string> s;

    for(int i = 0; i < words.size(); i++)
        ____________________;             //LINE-1
    while(!s.empty()){
        std::cout << __________ << " ";    //LINE-1
        ____________________;             //LINE-3
    }
}
int main() {
    int n;
    std::cin >> n;
    std::vector<std::string> vec;
    for(int i = 0; i < n; i++){
        std::string wd;
        std::cin >> wd;
        vec.push_back(wd);
    }
    printReverseOrder(vec);
    return 0;
}
```

### Public 1

```
Input: 4 Mercury Venus Earth Mars
Output: Mars Earth Venus Mercury
```

### Public 2

```
Input: 5 Arctic Antarctic Atlantic Pacific Indian
Output: Indian Pacific Atlantic Antarctic Arctic
```

### Private

```
Input: 7 Cow Sheep Dog Goat Horse Mouse Ox
Output: Ox Mouse Horse Goat Dog Sheep Cow
```

**Answer:**

```
LINE-1:  s.push(words[i])
LINE-2:  s.top()
LINE-3:  s.pop()
```

**Explanation:**

At `LINE-1`, each element can be added to the stack as:

```
s.push(words[i])
```

At `LINE-2`, the element at the top of the stack can be displayed as:

```
std::cout << s.top() << " ";
```

At `LINE-3`, the element at the top of the stack can be removed as:

```
LINE-3:  s.pop()
```