Programming in Modern C++: Assignment Week 9

Total Marks : 20

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

September 14, 2023

Question 1

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

```
#include<cstdio>

int main(){
    int i = 65;
    std::printf(____________________, i, i, i, i);    //LINE-1
    return 0;
}
```

Identify the correct option to fill in the blank at `LINE-1` such that the output becomes `65 41 101 A`.

a) `"%d %o %x %c"`

b) `"%d %x %o %c"`

c) `"%d %0x %o %s"`

d) `"%i %0x %o %s"`

Answer: b)
Explanation:
To print the value of `i` in decimal format, we use `%d`.
To print the value of `i` in hexadecimal format, we use `%x`.
To print the value of `i` in octal format, we use `%o`.
To print the value of `i` in char format, we use `%c`.

Question 2

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

```
#include <iostream>
#include <iomanip>

int main () {
    std::cout << std::setprecision(6) << std::setfill('0');
    std::cout << std::setw(8) << (double)10/3;
    return 0;
}
```

What will be the output?

a) `3.333333`

b) `03.33333`

c) `000003.33333333`

d) `03.333333`

Answer: b)
Explanation:
`10/3.0 = 3.3333...`
`setprecision(6)` makes it to `3.33333`
`setfill('0')` and `setw(8)` make it to `03.33333`
Hence, the correct option is b).

Question 3

Consider `fp` as a file pointer to a existing file, and the file is open in readonly mode. Match the appropriate descriptions about the `fseek` function calls. *[MCQ, Marks 2]*

Function calls	**Descriptions**
1. `fseek(fp, 100, SEEK_SET)`	A. Move the file pointer to the end of the file
2. `fseek(fp, -100, SEEK_CUR)`	B. Move the file pointer forward from the beginning of the file by 100 characters
3. `fseek(fp, 0, SEEK_END)`	C. Move the file pointer backwards from the current position in the file by 100 characters
4. `fseek(fp, -100, SEEK_END)`	D. Move the file pointer backwards from the end of the file by 100 characters.

a) `1-A, 2-D, 3-C, 4-B`

b) `1-A, 2-C, 3-D, 4-B`

c) `1-B, 2-A, 3-C, 4-D`

d) `1-B, 2-C, 3-A, 4-D`

Answer: d)
Explanation:

- `fseek(fp, 100, SEEK_SET)` move the file pointer forward from the beginning of the file by 100 positions.
- `fseek(fp, -100, SEEK_CUR)` moves the file pointer backwards from the current position in the file by 100 positions
- `fseek(fp, 0, SEEK_END)` moves the file pointer to the end of the file
- `fseek(fp, -100, SEEK_END)` moves the file pointer backwards from the end of the file by 100 positions

Question 4

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

```
#include<cstdio>

int main(){
    FILE *infp;
    if((infp = std::fopen("myfile.txt", "r")) == NULL)
        return -1;
    int c, n = 0;
    while((c = std::fgetc(infp)) != EOF)
        if(c == '\n' || c == ' ')      //LINE-1
            n++;
    std::printf("%d", n);
    fclose(infp);
    return 0;
}
```

Choose the correct option regarding the program.

a) It prints the number of blank spaces in file `myfile.txt`.

b) It prints the number of words in file `myfile.txt`.

c) It prints the number of lines in file `myfile.txt`.

d) It prints the number of characters in file `myfile.txt`.

Answer: b)
Explanation:
It increment `n` because (1) it counts on each blank space, (2) it counts on each newline, and (3) it counts on end of line. Therefore, it prints the number of words in file `myfile.txt`.

Question 5

Consider the code given below to print all the lines from `in.txt`. *[MCQ, Marks 2]*

```
#include <iostream>
#include <fstream>
#include <string>

int main () {
    std::ifstream infile("myfile.txt");
    std::string line;
    if (____________________) {    //LINE-1
        std::cout << "Unable to open file";
    }
    else{
        while (std::getline(infile, line))
            std::cout << line << std::endl;
        infile.close();
    }
    return 0;
}
```

Identify the appropriate option to fill in the blank at `LINE-1` such that it checks if the file does not exist.

a) `infile.is_open()`

b) `!infile.is_open()`

c) `!infile.open()`

d) `fopen(infile) == NULL`

Answer: b)
Explanation:
Since the program attempts to read from the file, if the file exists or not can be verified by `is_open()` function. Thus, the correct option is b).

Question 6

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

```
#include <iostream>
#include <list>
#include <numeric>

class purchase{
    public:
        purchase(int itm_no, double pri, int qty) : itm_no_(itm_no), pri_(pri),
                                                    qty_(qty){}
        double get_pri(){ return pri_; }
        int get_qty(){ return qty_; }
    private:
        int itm_no_;
        double pri_;
        int qty_;
};

struct compute_price{
    double operator()(double d, purchase p) {
        return d + p.get_pri() * p.get_qty();
    }
};


double compute_total(std::list<purchase> li, compute_price cp){
    double total = accumulate(_____________________________);    //LINE-1
    return total;
}

int main(){
    std::list<purchase> li { purchase(101, 500.0, 10), purchase(102, 400.0, 5),
                  purchase(102, 600.0, 5)};
    compute_price cp;
    double cost = compute_total(li, cp);
    std::cout << cost;
    return 0;
}
```

Which of the following statement can be used to fill in the blank at `LINE-1` that the output becomes `10000`.

a) `li.begin(), li.end()`

b) `li.begin(), li.end(), cp`

c) `li.begin(), li.end(), 0.0, cp`

d) `li.begin(), li.end(), 0.0`

Answer: c)
Explanation:
The arguments of `std::accumulate` function are starting and ending of list, the value to initialize the sum, and function (or functor) to be applied. Therefore, c) is the correct option.

Question 7

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

```
#include <iostream>
#include <algorithm>
#include <list>

int main() {
    char ca[] { 't', 'r', 'i', 'a', 'n', 'g', 'l', 'e' };
    int len = sizeof(ca) / sizeof(*ca);
    std::list<char> lc(len);

    ____________________________________________;    //LINE-1

    for(std::list<char>::iterator it = lc.begin(); it != lc.end(); ++it)
        std::cout << *it;
    return 0;
}
```

Identify the appropriate call to `copy` function to fill in the blank at `LINE-1` such that it prints `angle` as output.

a) `copy(ca, ca + len - 3, lc.begin())`

b) `copy(ca + 3, ca + len, lc.begin())`

c) `copy(&ca[2], &ca[len - 1], lc.begin())`

d) `copy(&ca[3], &ca[len], lc.begin())`

Answer: b), d)
Explanation:
The syntax of `copy` function is as follows:

```
template<class InputIterator, class OutputIterator>
OutputIterator copy(InputIterator first, InputIterator last, OutputIterator result)
```

The correct options are – b) and d).

Question 8

Consider the following code segment which computes inner product of the elements of list and vector. *[MSQ, Marks 2]*

```
#include <iostream>
#include <list>
#include <vector>
#include <numeric>

struct add{
    int operator()(int i, int j){ return i + j; }
};

struct multi{
    int operator()(int i, int j){ return i * j; }
};

int main() {
    std::list<int> li  { 4, 5, 6 };
    std::vector<int> vi  { 7, 8, 9 };

    int result = ________________________________________;    //LINE-1
    std::cout << result;
    return 0;
}
```

Identify the appropriate call/calls to `inner_product` function to fill in the blank at `LINE-1` such that it prints `122` as output.

a) `inner_product(li.begin(), li.end(), vi.begin(), 0, multi(), add())`

b) `inner_product(li.begin(), li.end(), vi.begin(), 0, add(), multi())`

c) `inner_product(vi.begin(), vi.end(), li.begin(), 0.0, add(), multi())`

d) `inner_product(vi.begin(), vi.end(), li.begin(), 0.0, multi(), add())`

Answer: b), c)
Explanation:
The code by `inner_product` function is:

```
template<class In, class In2, class T, class BinOp, class BinOp2 >
T inner_product(In first, In last, In2 first2, T init, BinOp op, BinOp2 op2) {
    while(first!=last) {
        init = op(init, op2(*first, *first2));
        ++first; ++first2;
    }
    return init;
}
```

Thus, b) and c) are the correct option.

Question 9

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

```
#include<iostream>
#include <algorithm>
#include<vector>

class product{
    public:
        product(int prod_num, std::string prod_nam) :
                                prod_num_(prod_num), prod_nam_(prod_nam){}
        int get_prod_num(){ return prod_num_; }
        std::string get_prod_nam(){ return prod_nam_; }
    private:
        int prod_num_;
        std::string prod_nam_;
};

struct compare{
    bool operator()(product p1, product p2){
        if(p1.get_prod_nam() == p2.get_prod_nam()){
            if(p1.get_prod_num() > p2.get_prod_num()){
                return true;
            }
            else{
                return false;
            }
        }
        else if(p1.get_prod_nam() > p2.get_prod_nam())
            return true;
        else
            return false;
    }
};
int main() {
    std::vector<product> prod { product(103, "Pen"), product(102, "Pencil"),
                                    product(106, "scale"), product(104, "Pen"),
                                    product(105, "sharpener") };
    sort(prod.begin(), prod.end(), compare());
    for(std::vector<product>::iterator it = prod.begin(); it < prod.end(); it++)
        std::cout << it->get_prod_num() << " : " << it->get_prod_nam() << std::endl;
    return 0;
}
```

What will be the output?

a)
```
105 : sharpener
106 : scale
102 : Pencil
104 : Pen
103 : Pen
```

b)
```
105 : sharpener
```

```
106 : scale
102 : Pencil
103 : Pen
104 : Pen
```

c)
```
103 : Pen
104 : Pen
102 : Pencil
106 : scale
105 : sharpener
```

d)
```
104 : Pen
103 : Pen
102 : Pencil
106 : scale
105 : sharpener
```

Answer: a)
Explanation: The implementation of functor `compare` sorts the `product` objects in descending order by the name (`prod_nam_`) of the `product`s. If any two `product` objects have same name, then it will be sorted in descending order by the product number (`prod_num_`) of the students. Thus, a) is the correct option.

Programming Questions

Question 1

Consider the following program that finds the minimum element in an array. Fill in the blanks as per the instructions given below:

- Fill in the blank at `LINE-1` with appropriate template declaration.
- Fill in the blanks at `LINE-2` and `LINE-3` with appropriate conditional statement.

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

```
#include<iostream>

_____________________________________    //LINE-1
void min(Itr first, Itr last, T& mv) {
    mv = *first++;
    while (___________________) {          //LINE-2
        if(___________________)            //LINE-3
            mv = *first;
        ++first;
    }
}

int main(){
    int iArr[10];
    int n;
    std::cin >> n;
    for(int i = 0; i < n; i++)
        std::cin >> iArr[i];

    double minVal = 0.0;
    min(iArr, iArr + n, minVal);
    std::cout << minVal;
    return 0;
}
```

Public 1

```
Input:
5
50 27 10 70 23
Output: 10
```

Public 2

```
Input:
6
94 5 22 45 32 1
Output: 1
```

Private

```
Input:
4
10 20 40 30
Output: 10
```

Answer:

```
LINE-1:  template<typename Itr, typename T>
```

or

```
LINE-1:  template<class Itr, class T>
LINE-2:  first != last
LINE-3:  *first < mv
```

Explanation:

At `LINE-1` we can declare the template as:

```
template<typename Itr, typename T>
```

or `template<class Itr, class T>`

At `LINE-2` we have to provide a condition to check the end of the array, which can be written as:

```
first != last
```

At `LINE-3` we have to check whether `mv` is still the minimum value or not, which can be written as: `*first < mv`

Question 2

Consider the following program that takes inputs a lower limit (`l`) and an upper limit (`u`) of a vector. If all the elements of the input vector are within the given lower and upper limits, the program prints "all in [l, u]". Otherwise, the program prints the first element of the vector which is not within the given limits. Fill in the blanks as per the instructions given below:

- at `LINE-1` with appropriate header to overload function operator,
- at `LINE-2` with appropriate template definition,
- at `LINE-3` with appropriate condition,

such that the program will satisfy the given test cases. *Marks: 3*

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

struct Bound{
    Bound(int l, int u) : l_(l), u_(u){}
    ____________________________{     //LINE-1
        return (n >= l_ && n <= u_);
    }
    int l_, u_;
};

_______________________________      //LINE-2
Itr search(Itr first, Itr last, Pred bd) {
    while (first != last){
        if(____________________)     //LINE-3
            return first;
        ++first;
    }
    return first;
}

int main(){
    int iArr[] = { 70, 20, 50, 40, 90, 10, 80 };
    std::vector<int> iVec(iArr, iArr + sizeof(iArr) / sizeof(*iArr));
    int l, u;
    std::cin >> l >> u;
    Bound bd(l, u);

    std::vector<int>::iterator it = search(iVec.begin(), iVec.end(), bd);

    if(it == iVec.end())
        std::cout << "all in [" << l << ", " << u << "]";
    else
        std::cout << *it << " is the first element not in [" << l << ", " << u << "]";
    return 0;
}
```

Public 1

```
Input: 10 100
```

```
Output: all in [10, 100]
```

Public 2

```
Input: 10 70
Output: 90 is the first element not in [10, 70]
```

Private

```
Input: 40 90
Output: 20 is the first element not in [40, 90]
```

Answer:

```
LINE-1:  bool operator()(int n)
LINE-2:  template<class Itr, class Pred>
```

or

```
LINE-2:  template<typename Itr, typename Pred>
LINE-3:  !bd(*first)
```

Explanation:

At `LINE-1`, the header to overload function operator is `bool operator()(int n)`.

At `LINE-2`, the template for the `search()` function can be written as:

```
template<class Itr, class Pred>
```

or

```
template<typename Itr, typename Pred>
```

At `LINE-3`, the `if` condition can be written as:

```
if(!bd(*first))
```

Question 3

Consider the following program, which computes the frequency of occurrence (histogram) of each integer in a given vector. Fill in the blanks as per the instructions given below:

- at `LINE-1` with appropriate statement to iterate over the given vector `v`,
- at `LINE-2` with appropriate statement to iterate over the given map `hi`,

such that it will satisfy the given test cases. *Marks: 3*

```
#include <iostream>
#include <map>
#include <vector>

std::map<int, int> histo(std::vector<int> v){
    std::map<int, int> hi;
    for (_____________________________________________)   //LINE-1
        hi[*it]++;
    return hi;
}

void print(std::map<int, int> hi){
    for (_____________________________________________)   //LINE-2
        std::cout << it->first << ": " << it->second << ", ";
}

int main() {
    std::vector<int> vec;
    for(int i = 0; i < 10; i++){
        int a;
        std::cin >> a;
        vec.push_back(a);
    }
    std::map<int, int> hi = histo(vec);
    print(hi);
    return 0;
}
```

Public 1

```
Input: 9 4 5 3 4 6 7 4 3 5
Output: 3: 2, 4: 3, 5: 2, 6: 1, 7: 1, 9: 1,
```

Public 2

```
Input: 6 6 7 3 4 5 6 5 4 7
Output: 3: 1, 4: 2, 5: 2, 6: 3, 7: 2,
```

Private

```
Input: 1 3 4 5 3 3 2 1 4 3
Output: 1: 2, 2: 1, 3: 4, 4: 2, 5: 1,
```

Answer:

```
LINE-1:  std::vector<int>::iterator it = v.begin(); it != v.end(); ++it
LINE-2:  std::map<int,int>::iterator it = hi.begin(); it != hi.end(); ++it
```

Explanation:

The `std::vector<int>::iterator` type can be used to iterate through the given vector `v`, so the statement at `LINE-1` must be

```
for(std::vector<int>::iterator it = v.begin(); it != v.end(); ++it
```

Similarly, the `std::map<char,int>::iterator` type can be used to iterate through the given map `hi`, so the statement at `LINE-2` must be

```
for (std::map<int,int>::iterator it = hi.begin(); it != hi.end(); ++it)
```