Roll No _____ Block No _____

Make Up Quiz

General instructions: Time: 2 hours

- There are 4 questions in this exam. Write your answers directly on this paper in the spaces provided.
- Do not use a more general type where a simpler type would work (e.g., do not use a **float** type where **int** will work.)
- Follow C++ syntax **strictly** when completing missing parts of code.

Q1).**[5 Marks]**

(a).Write the output of the following code shown below.

```
#include <simplecpp>
bool f(int a[], int beg, int end, int item)
{
        if(beg == end)
        {
           if(item == a[beg]) return true;
           else return false;
        }
        else
        {
           int mid = (beg + end)/2;
        if(item == a[mid])  return true;
           else if(item < a[mid])  return f(a,beg,mid-1,item);
         else return f(a,mid+1,end,item);
     }
}
```

```
int main() {
```

Roll No _____ Block No _____

```
        int a[5] = {1,2,4,6,7};
        int b[5] = {4,6,8,10,12};
        cout<<f(a,0,4,2)<<endl;
        cout<<f(b,0,4,7)<<endl;
        return 0;
    }
```

Answer:

_____

_____

(b). Given that the array a[ ] (size of a = n) argument to the function is sorted in ascending order, explain what does function f(a,0,n-1,x) do? In particular, explain on what conditions on parameters does the function f(a,0,n-1,x) does the function returns true?

Answer:

_____

_____

_____

Solution to part a:

1

0

Solution to part b:

The function f searches for the item x in the entire array a. Specifically, f returns true if there exists at least one i such that 0 <= i <= n - 1 and a[i] = x.

Marks distribution:

3+2 = 5

Roll No _____ Block No _____

Q2). **[12 Marks]** Search engines like Google, Bing, etc. crawl the web and store certain information about websites that is used to rank the websites when a search query is given by the user.

Following is a C++ struct storing some relevant information about a webpage such as:

i. ID: a 32-bit signed integer value
ii. URL: the web address of the webpage
iii. Keywords: a vector of strings (words) relevant to the webpage
iv. Hyperlinks: a vector of IDs of other webpages to which this webpage has links to

Template:

```
struct Webpage {
        int id;
        string url;
        vector<string> keywords;
        vector<int> hyperlinks;
};

vector<Webpage> webpages; // global vector of webpages
```

(a).**Count total number of inlinks**: The **hyperlinks** vector only stores the number of "outgoing" links from the given webpage. Following is a template of a function that computes the total number of webpages that have at least one hyperlink to the given webpage. Complete the template as instructed in the comments:

```
int countInlinks(Webpage w) {
        // assume that the vector webpages is accessible in this scope

        int count = 0;
        for (int i = 0; i < _____; i++) {
         // iterate over all webpages

                if (_____) {
                // no need to count hyperlink to itself

                        bool found = false;

                        for(int j = 0; j < _____; j++) {
```

```
                    if (_____) {
                        found = true;
                        break;
                    }
                }
                if (found) count++;
            }
        }
        return count;
}
```

Answer (a):

```
int countInlinks(Webpage w) {
        // assume that the vector webpages is accessible in this scope
        int count = 0;
        for (int i = 0; i < webpages.size(); i++) {
            if (w.id != webpages[i].id) {
                bool found = false;
                for(int j = 0; j < webpages[i].hyperlinks.size(); j++) {
                    if (webpages[i].hyperlinks[j] == w.id) {
                        found = true;
                    }
                }
                if (found) count++;
            }
        }
        return count;
}
```

(b).**Count number of matching keywords with a given query**: Given a search query (a vector of strings), following is a template of a function that counts the number of keywords associated with the webpage that are same as the words in the query. Complete the template as instructed in the comments:

```
int countMatchingKeywords(Webpage w, vector<string> query) {
        int count = 0;
        for(int i = 0; i < _____; i++) {
```

```
            for(int j = 0; j < _____; j++) {
                if (_____)
                    count++;
            }
        }
        return count;
}
```

Note: Assume that all elements of vector<string> query and vector<string> keywords (for Webpage w), are distinct.

Answer (b):

Two solutions are possible:

Solution 1:

```
int countMatchingKeywords(Webpage w, vector<string> query) {
        int count = 0;
        for(int i = 0; i < query.size(); i++) {
            for(int j = 0; j < w.keywords.size(); j++) {
                if (query[i] == w.keywords[j])
                    count++;
            }
        }
        return count;
}
```

Solution 2:

```
int countMatchingKeywords(Webpage w, vector<string> query) {
        int count = 0;
        for(int i = 0; i < w.keywords.size(); i++) {
            for(int j = 0; j < query.size(); j++) {
                if (query[j] == w.keywords[i])
                    count++;
            }
        }
        return count;
}
```

(c). **Searching for a query**: Given a query(which is a vector of strings), find the webpage that is most relevant for that query. The most relevant result is the webpage with maximum

value of number of inlinks(as determined in part a) multiplied by the number of matching keywords with the query(as determined in part b):

```
Webpage getMostRelevantWebpage(vector<string> query) {
        // assume that the vector webpages is accessible in this
        scope and that it has a size of at least 1

    int maxx = countInlinks(webpages[0]) *
            countMatchingKeywords(webpages[0], query);
  Webpage mostRelevant = webpages[0];
    for(int i = 1; i < _____; i++) {
        if(_____) {

          _____;
          _____;
      }
    }
    return _____;
}
```

Solution:

```
Webpage best_result(vector<string> query) {
        // assume that the vector webpages is accessible in this
        scope and that it has a size of atleast 1

    int maxx = countInlinks(webpages[0]) *
            countMatchingKeywords(webpages[0], query);
  Webpage most_relevant = webpages[0];
    for(int i = 1; i < webpages.size(); i++) {
        if(countInlinks(webpages[i])  *
                countMatchingKeywords(webpages[i], query) >
                maxx)
        {

         // Note: order of the following 2 statements can be
interchanged
              maxx = countInlinks(webpages[i]) *
                    countMatchingKeywords(webpages[i], query);
```

```
            most_relevant = webpages[i];
        }
    }
    return most_relevant;
}
```

Todo:

Add evaluation instruction/marks for partial answers etc.

Marks

1 mark for each blank
Total: 12 marks

Roll No _____ Block No _____

(Q3). **[8 Marks]** Given below is a code to perform matrix multiplication. The first matrix M1 has size **r1 X c1** i.e., ‘r1’ rows and ‘c1’ columns, while the second matrix M2 has size **r2Xc2** i.e., ‘r2’ rows and ‘c2’ columns. The product matrix M3 has r1 rows and c2 columns. Now fill the blanks in the following code.

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

using namespace std;

int main(){
  int r1 = 3, c1 = 2, r2 = 2, c2 = 3;
  vector<vector<int> > M1(r1, vector<int>(c1));
  for(int i=0; i<r1; i++) {
     for(int j=0; j<c1; j++) {
        cin >> M1[i][j];
     }
  }
  vector<vector<int> > M2(r2, vector<int>(c2));
  for(int i=0; i<r2; i++) {
     for(int j=0; j<c2; j++) {
        cin >> M2[i][j];
     }
  }
  if(____ != ____) {
     cout << "Matrix Multiplication not possible\n";
     return 0;
  }
  vector<vector<int> > M3(____, vector<int>(____, 0));
```

Roll No _____ Block No _____

```
  for(int i=0; i<r1; i++) {
     for(int j=0; j<c2; j++) {
        for(int k=0; k<c1; k++) {
           M3[i][j] += _____ * ____;
        }
     }
  }

  for(int i=0; i<r1; i++) {
     for(int j=0; j<c2; j++) {
        cout << M3[i][j] << " ";
     }
     cout << endl;
  }
}
```

Solution:

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

using namespace std;

int main(){
  int a = 3, b = 2, m = 2, n = 3;
  vector<vector<int> > M1(a, vector<int>(b));
  for(int i=0; i<a; i++) {
     for(int j=0; j<b; j++) {
        cin >> M1[i][j];
     }
  }
  vector<vector<int> > M2(m, vector<int>(n));
  for(int i=0; i<m; i++) {
     for(int j=0; j<n; j++) {
```

Roll No _____ Block No _____

```
            cin >> M2[i][j];
        }
    }
    if( b != m ) {
        cout << "Not possible\n";
        return 0;
    }
    vector<vector<int> > M3( a , vector<int>( n , 0));

    for(int i=0; i<a; i++) {
        for(int j=0; j<n; j++) {
            for(int k=0; k<b; k++) {
                M3[i][j] += V1[i][k] * V2[k][j];
            }
        }
    }

    for(int i=0; i<a; i++) {
        for(int j=0; j<n; j++) {
            cout << M3[i][j] << " ";
        }
        cout << endl;
    }
}
```

Marks

1 mark for first 4 blanks. 2 marks for other two.
Total: 8 marks

Roll No _____ Block No _____

(Q4). **[5 Marks]** Fill the below code which finds the number of common integers between two arrays. Assume the integers lie between 0 and 99 inclusive and to be non-repetitive within an array.

```
#include <simplecpp>
int common(int a1[], int a2[], int n1, int n2){
        int count[100];
        int num_common = 0;
        for (int i = 0; i < 100; ++i)
        // initialising the count array to zero
        {
                _______= 0;
        }

        for (int i = 0; i < n1; ++i)
        // Finding the integers present in first array
        {
               count[_____] = _______;
        }

        for (int i = 0; i < n2; ++i)
        // Counting the number of common integers between first and
           second array.
        {
                if(count[_______] == 1)    _______++;
        }

        return num_common;
}

main_program{
        int n1,n2;
        cin >> n1 >> n2;
        int a1[n1], a2[n2];

        for (int i = 0; i < n1; ++i)
```

```
        {
                cin >> a1[i];
        }
        for (int i = 0; i < n2; ++i)
        {
                cin >> a2[i];
        }

        cout << common(a1,a2,n1,n2);
}
```

Solution Code:

```
#include <simplecpp>

int common(int a1[], int a2[], int n1, int n2){
        int count[100];
        int num_common = 0;
        for (int i = 0; i < 100; ++i) // initialising the count array to zero
        {
                count[i] = 0;
        }

        for (int i = 0; i < n1; ++i) // Finding the integers present in first array
        {
                count[a1[i]] = 1;
        }

        for (int i = 0; i < n2; ++i) // Counting the number of common integers between first and second array.
        {
                if(count[a2[i]] == 1)  num_common++;
        }

        return num_common;
}
```

Roll No _____ Block No _____

```
main_program{
        int n1,n2;
        cin >> n1 >> n2;
        int a1[n1], a2[n2];
        for (int i = 0; i < n1; ++i)
        {
                cin >> a1[i];
        }
        for (int i = 0; i < n2; ++i)
        {
                cin >> a2[i];
        }

        cout << common(a1,a2,n1,n2);
}
```

Marks

- 1 mark each.
- Total: 5 marks