National University of Computer and Emerging Sciences, Lahore Campus

	Course Name:	**Object Oriented Programming**	**Course Code:**	**CS217**
	Degree Program:	**BS (CS, SE, DS)**	**Semester:**	**Spring 2022**
	Exam Duration:	**180 Minutes**	**Total Marks:**	**65**
	Paper Date:	**13-June-2022**	**Weight**	**40**
	Section:	**ALL**	**Page(s):**	**13**
	Exam Type:	**Final Exam**		

Student : Name:_____ **Roll No.**_____ **Section:**_____

Instruction/Notes: Attempt all questions. Answer in the space provided. **Answers written on rough sheet will not be marked**. Do not use pencil or red ink to answer the questions**.** In case of confusion or ambiguity make a reasonable assumption. Properly comment your code in Q1, Q2 and Q3.

Question 1: (CLO:3,4) **(Marks: 15)**

Suppose you are working as an intern in a large company that does a lot of text processing and your team lead wants you to write a small function that takes an array of tweets as input and create an index of important key words in all the tweets. According to the guidelines given by your boss, important key words are all the words separated by white space. For simplicity you can assume that all the letters are in lower case

Consider the following example:

Tweet 1 breakthrough drug schizophrenia drug released July
Tweet 2 new schizophrenia drug breakthrough drug
Tweet 3 new approach treatment schizophrenia
Tweet 4 new hopes schizophrenia patients schizophrenia cure

Important key words are: breakthrough, drug, schizophrenia, released, july, new, approach, treatment, hopes, patients, cure

An index contains key words and for each key word it stores the tweet id in which that key word appears. Below is an example of an index

breakthrough	1	2		
drug	1	2		
schizophrenia	1	2	3	4
released	1			
july	1			
new	2	3	4	
approach	3			
treatment	3			
hopes	4			
patients	4			
cure	4			

Write a C++ function ***createIndex*** that takes an array of strings (char**) named ***tweets*** and its size ***n*** as parameters. This function will first extract all the unique key words and store them in a dynamically created array of strings(***char** keyWords***). It must also create another dynamic 2D array "***IDs***" of integers such that for each key word in the array ***keyWords*** there is a row that stores the tweet ids of all those tweets in which that word appears, followed by -1. The size of the row must be precisely equal to the number of tweets in which that key word appears + 1. You also have to return both the arrays: ***keyWords, IDs*** and their ***size*** from the function ***createIndex*****.**

You can divide the tasks into smaller subtasks. You can also use built-in functions for string processing.

Sample Output:

keyWords
breakthrough
Drug
schizophrenia
Released
July
New
Approach
Treatment
Hopes
Patients
Cure

IDs

1	2	-1		
1	2	-1		
1	2	3	4	-1
1	-1			
1	-1			
2	3	4	-1	
3	-1			
3	-1			
4	-1			
4	-1			
4	-1			

```
template<class T>
void grow(T** &arr, int n) {
	T** newarr = new T*[n + 1];
	for (int i = 0;i < n;i++)
		newarr[i] = arr[i];
	delete[]arr;
	arr = newarr;
}
void grow(int *& arr, int n) {
	int * newarr = new int [n + 1];
	for (int i = 0;i < n;i++)
		newarr[i] = arr[i];
	delete[]arr;
	arr = newarr;
}
```

```
void createIndex(char** tweets, int n, char**& keywords, int **& IDs, int & size){
	char delimeter[] = " ";
	char* tok;
	size = 0;
	int* IDs_size = new int[0];
	keywords = new char* [0];
	IDs = new int* [0];
	bool found;
	//iterate for all tweets
	for (int k = 0;k < n;k++) {
		// get space seperated keywords in each tweet
		tok = strtok(tweets[k], delimeter);
		while (tok) {
			found = false;
			int ind = 0;
			//check if the key word already dound or not
			for (int check= 0;check < size;check++) {
				if (strcmp(tok, keywords[check]) == 0) {
					found = true;
					ind = check;
				}
			}
			//if keyword not found then grwo the keywords IDs array and update
			if (!found) {
				grow(keywords, size);
				grow(IDs, size);
				grow(IDs_size, size);
				keywords[size] = new char[strlen(tok) + 1];
				strcpy(keywords[size], tok);
				IDs_size[size] = 2;

				IDs[size] = new int[IDs_size[size]];
				IDs[size][0] = k+1;
				IDs[size][1] = -1;
				size++;
			}
			else {//if key word already found
				int x;
				//check for multiple occurrence in the tweet
				for (x = 0; x < IDs_size[ind] - 1; x++)
					if (IDs[ind][x] == k + 1)
						break;
				//if found in new tweet then add tweet ID
				if (x == IDs_size[ind] - 1) {
					grow(IDs[ind], IDs_size[ind]);
					IDs[ind][IDs_size[ind] - 1] = k + 1;
					IDs[ind][IDs_size[ind]] = -1;
					IDs_size[ind]++;
				}
			}
			tok = strtok(NULL, delimeter);

		}

	}
}
```

Question 2 (CLO:3,4) **(Marks: 15)**

A digital image is a representation of a real image which is stored in a matrix format. Each cell of matrix is called pixel (picture element) which contains an integer value. Your task is to provide the functionality for class image that includes: **implementation of default and parameterized constructor, destructor and a public method (filtering).** Consider the partial definition of class Image.

```
class Image
{
   int rows, cols;
   int **img;
   public:
        … // Complete the code
};
```

1. Create default constructor [2]

```
Image()
  {
    rows=0;
    cols=0;
    img = nullptr;
  }
```

2. Create a parameterized(overloaded) constructor that takes rows, columns, and a 2D pointer **img_ptr** to a 2D array as parameters and dynamically create the array **img** and initialize **img** to img_ptr. [3]

```
Image(int **img_ptr, int rows,int cols)
  {
    int i=0,j=0;
    this->rows=rows;
    this->cols=cols;
    img = new int*[rows];
    for(; i<rows; i++)
      img[i]=new int[cols];
    for(i=0; i<rows; i++)
      for(j=0; j<cols; j++)
        img[i][j]=img_ptr[i][j];
}
```

3. Destructor [2]

```
~Image()
  {
    for(int i=0; i<rows; i++)
      delete img[i];
    delete img;
  }
```

4. Create a Function Filtering which takes a 2D matrix named **filter** of size **mxn** and an integer stride as parameters, and return the filtered image by following the steps given below: [8]
   a. Calculate *divisor* by adding all the values in the filter.
   b. Slide the *filter* onto the image by the jump as per *stride(jump in both directions row and column)*.

c. Multiply the corresponding elements of filter and image and then add them. After that divide the answer by *divisor* and get new value.
d. Repeat this procedure until all values of the image has been calculated.
e. The new value should be placed in a new 2D array which will be returned from the function. The calculated value should be placed at the exact center point where the filter is applied. To ensure an exact center point, the size of filter must be in odd number. (you can assume it's odd in number)
f. The size of the new filtered image and input image should remain the same. To accomplish it the boundaries should have the old values as per the original input image.

Example: If filter is applied on the part that is within the black boundary of the image given above, then the updation would be as follows:

Divisor = 0+1+0+1+2+1+0+1+0=6

The values in image where *filter* will be updated [47, 22, 25, 52, 51, 50, 35, 47, 49]

Corresponding values in *filter* [0 , 1 , 0 , 1 , 2 , 1 , 0 , 1 , 0]

New Value = [47×0 + 22×1 + 25×0 + 52×1 + 51×2 + 50×1 + 35×0 + 47×1 + 49×0] / 6 = 45

The new value, which is 45, should be updated in the resultant new image.

Image

35	50	45	46	67
36	47	22	25	29
38	52	51	50	54
42	35	47	49	51
53	62	63	66	70

Filter

0	1	0
1	2	1
0	1	0

Filtered Image

35	50	45	46	67
36	47	22	25	29
38	52	**45**	50	54
42	35	47	49	51
53	62	63	66	70

Complete Example for stride = 2

35	50	45	46	67
36	47	22	25	29
38	52	51	50	54
42	35	47	49	51
53	62	63	66	70

0	1	0
1	2	1
0	1	0

35	50	45	46	67
36	**42**	22	25	29
38	52	51	50	54
42	35	47	49	51
53	62	63	66	70

35	50	45	46	67
36	47	22	25	29
38	52	51	50	54
42	35	47	49	51
53	62	63	66	70

0	1	0
1	2	1
0	1	0

35	50	45	46	67
36	47	22	**32**	29
38	52	51	50	54
42	35	47	49	51
53	62	63	66	70

35	50	45	46	67
36	47	22	25	29
38	52	51	50	54
42	35	47	49	51
53	62	63	66	70

0	1	0
1	2	1
0	1	0

35	50	45	46	67
36	47	22	25	29
38	52	51	50	54
42	**45**	47	49	51
53	62	63	66	70

35	50	45	46	67
36	47	22	25	29
38	52	51	50	54
42	35	47	49	51
53	62	63	66	70

0	1	0
1	2	1
0	1	0

35	50	45	46	67
36	47	22	25	29
38	52	51	50	54
42	35	47	**52**	51
53	62	63	66	70

Filtered Image:

35	50	45	46	67
36	**42**	22	**32**	29
38	52	51	50	54
42	**45**	47	**52**	51
53	62	63	66	70

```
int** filtering(int **filter,int m,int n, int stride=1)
  {
     int i=0,j=0,k=0,l=0,divisor=0,val=0,m2,n2;
     int **filtered_img = new int*[rows];
     for(i=0; i<rows; i++)
        filtered_img[i]=new int[cols];
    copyBoundary(img,filtered_img,m,n,stride);
     for(i=0;i<m;i++)
        for(j=0;j<n;j++)
           divisor+=filter[i][j];
      for(i=m/2;i<(rows-m/2);i+=stride)
        for(j=n/2;j<(cols-n/2);j+=stride)
        {
           val=0;
           m2=-m/2;
           for(k=0;k<m;k++)
           {
              n2=-n/2;
              for(l=0;l<n;l++)
                 val+=img[i+m2][j+n2++]*filter[k][l];
              m2++;
           }
           filtered_img[i][j]=val/divisor;
        }
     return filtered_img;
  }
  void copyBoundary(int **img1, int **img2,int m, int n,int stride=1) // or copy image
  {
     int endRows,endCols;
     m/2>stride ? endRows=m/2 :endRows=stride;
     n/2>stride ? endCols=n/2 :endCols=stride;
     cout<<endCols<<endl;
     cout<<endRows<<endl;
     for(int i=0; i<rows; i++)
           if(i<m/2 || i>=rows-(endRows))
              for(int j=0; j<cols; j++)
                 img2[i][j] = img1[i][j];

        for(int j=0; j<cols; j++)
           if(j<n/2 || j>=cols-(endCols))
              for(int i=0; i<rows; i++)
                    img2[i][j] = img1[i][j];
  }
```

Question 3 (CLO:2) **(Marks: 15)**

Consider a calendar management application that can track events (e.g. course lecture, workshop, seminar, recruitment drive, etc.). Every event has a title and date. Some events are recurring events that repeat based upon frequency (such as daily meeting, weekly lecture, etc). The goal is to handle and find occurrence dates of such recurring events using inheritance and polymorphism. The class definitions of **Date, Event** and **RecurringEvent** are given to you. You need to work out **DailyEvent, WeeklyEvent and MonthlyEvent** class definitions as well as proper implementation of necessary functions in order to satisfy the given main and achieve the output as illustrated below.

```
class Date
{
   protected:
      int year;
      int month;
      int day;

   public:
      Date (int,int,int);
      void addDay(int);
      void addMonth(int);
      void addYear(int);
      void print();
};
```

```
class Event
{
   protected:
      string title;
      Date date;

   public:
      Event(string,const Date&);
      virtual void print();
};

class RecurringEvent : public Event
{
  public:
     RecurringEvent(string,const Date&);
     virtual Event** occurrences(int) = 0;
};
```

```
int main(){
   RecurringEvent** rec = new RecurringEvent*[3];
   rec[0] = new DailyEvent("client meeting",Date(2022,06,08));
   rec[1] = new WeeklyEvent("lecture",Date(2022,06,08));
   rec[2] = new MonthlyEvent("project monitoring",Date(2022,06,08));

   for (int i=0; i < 3; i++){
      Event** occurrences = rec[i]->occurrences(3);
      for(int j=0; j < 3; j++){
          occurrences[j]->print();
          cout << endl;
      }
   }

   // …
   return 0;
}
```

Output:

client meeting: 2022-06-08; client meeting: 2022-06-09; client meeting: 2022-06-10
lecture: 2022-06-08; lecture: 2022-06-15; lecture: 2022-06-22
project monitoring: 2022-06-08; project monitoring: 2022-07-08; project monitoring: 2022-08-08;

Provide your solution for each class within the space given below:

```
// DailyEvent class definition and implementation


class DailyEvent: public RecurringEvent{
public:
DailyEvent(string t,const Date& d): RecurringEvent(t,d){}
virtual Event** occurrences(int);
};
Event** DailyEvent::occurrences(int c){
Event** inst = new Event*[c];
Date d = date;
for(int i=0; i < c; i++ ){
inst[i] = new Event(title,d);
d.addDay(1);
}
return inst;
}
```

```
// WeeklyEvent class definition and implementation

class WeeklyEvent: public RecurringEvent{
public:
WeeklyEvent(string t,const Date& d): RecurringEvent(t,d){}
virtual Event** occurrences(int);
};
Event** WeeklyEvent::occurrences(int c){
Event** inst = new Event*[c];
Date d = date;
for(int i=0; i < c; i++ ){
inst[i] = new Event(title,d);
d.addDay(7);
}
return inst;
}
```

```
// MonthlyEvent class definition and implementation

class MonthlyEvent: public RecurringEvent{
public:
MonthlyEvent(string t,const Date& d): RecurringEvent(t,d){}
virtual Event** occurrences(int);
};
Event** MonthlyEvent::occurrences(int c){
Event** inst = new Event*[c];
Date d = date;
for(int i=0; i < c; i++ ){
inst[i] = new Event(title,d);
d.addMonth(1);
}
return inst;
}
```

Question 4 (CLO:1) **(Marks: 5*4)**

Part a. Show the output of the given code.

```
class A
{
   public:
        virtual void funA(){
           cout<<"funA"<<endl;}
        void funB(){
           cout<<"funB"<<endl; }
        void funC (){
           cout<<"funC"<<endl; }
        void funD(){
           cout<<"funD"<<endl;
           funC();
           funA(); }
        virtual ~A(){
           cout<<"Virtual Destructor -ClassA"<<endl; }
};
```

```
class B:public A
{
   public:
   void funA(){
      cout<<"funA-ClassB"<<endl;   }
   void funB() {
      cout<<"funB-ClassB"<<endl;  }
     ~B() {
      cout<<"Destructor -ClassB"<<endl;    }
};
int main()
{
   A *aptr=new B;
   aptr->funA();
   aptr->funB();
   aptr->funC();
   aptr->funD();
   delete aptr;
     return 0;
}
```

Output:

funA-ClassB
funB
funC
funD
funC
funA-ClassB
Destructor -ClassB
Virtual Destructor -ClassA

Part b. There is no syntax error in this code. Determine the output

```
class addressType{
    string address;
    string city;
    string state;
    int ZIP_code;
public:
    addressType(){
        cout<<"Address storage"<<endl;
        address="pak";
        city="Lahore";
        state="11unjab";
        ZIP_code=54000;}
   void set_address(){
       address = "123 abc xyz";
        city = "Lahore";
        state = "Punjab";
        ZIP_code = 54000; }
   void display(){
       cout<<address<<" " ;
        cout<<city<<" ";
        cout<<state<<" ";
        cout<<ZIP_code<<endl;}
 ~addressType(){
       cout<<"Ending"<<endl; }
};

Class Person
{
  string name;
  string lastname;
  addressType *addr[3];
  public:
  Person(){
      name="Anthony";
      lastname="Lance";
      for (int i=0;i<3;i++)
           addr[i]=new addressType();
       }
  void setter(){
       name="Tim" ;
       lastname="white" ;}
  void display(){
       for (int i=0;i<3;i++)
          addr[i]->display() ;
        cout<<name;
        cout<<" "<<lastname<<endl;}
   ~Person(){
        for (int i=0;i<3;i++){
           delete addr[i] ;
           cout<<i<<" deleted"<<endl;}
   }
};
```

```
void sett(Person **pp)
{
    for(int i=0;i<2;i++)
    {
         if(i==1)
        {
            pp[i]->setter();
            pp[i]->display();
            return;
        }
        pp[i]->display();
    }
}
int main()
{  Person *p[2];
   for(int i=0;i<2;i++)
   {
    p[i]=new Person();
   }
   sett(p);
  for (int i=0;i<2;i++)
       {
           delete p[i] ;
           cout<<i<<" deleted"<<endl;
        }
}
```

Output:

Address storage
Address storage
Address storage
Address storage
pak Lahore Punjab 54000
pak Lahore Punjab 54000
pak Lahore Punjab 54000
Anthony Lance
pak Lahore Punjab 54000
pak Lahore Punjab 54000
pak Lahore Punjab 54000
Tim white
Ending
0 deleted
Ending
1 deleted
Ending
2 deleted
0 deleted
Ending
0 deleted
Ending
1 deleted
Ending
2 deleted
1 deleted

Part c: Partial output of the code is given in right column, write the output in rows with question mark? Assume that a class Person is already defined which is not a template class.

Code	Output
`#include "Person.h";// it' s a fully functional class`	

`//template function`
`template <class T>`
`void my_swap(T &one, T &two)`
`{`
`  T temp = one;`
`  one = two;`
`  two = temp;`
`  cout << "Swap successful";`
`}`	

`template <>`
`void my_swap(Person &one, Person &two)`
`{`
`  cout << "You cannot swap Person ";`
`}`	

`int main()`	
`{`	
`  int a=10, b=20;`	
`  cout <<a<<" "<<b<<endl;`	10 20
`  my_swap(a,b);` ?	Swap successful
`  cout <<a<<" "<<b<<endl;` ?	20 10
`  double *x= new double(10.5);`	
`  double *y= new double(11.5);`	
`  cout <<*x<<" "<<*y<<endl;`	10.5 11.5
`  my_swap(*x,*y);` ?	Swap successful
`  cout <<*x<<" "<<*y<<endl;` ?	11.5 10.5
`  //overloaded constructor takes account name as input`	
`  Person P1("Ron"), Person P2("Harry");`	
`  cout<<P1<<" "<<P2;`	Ron Harry
`  my_swap(P1, P2);` ?	You cannot swap person
`  cout<<P1<<" "<<P2;` ?	Ron Harry
`}`	
How many instances (copies) of my_swap functions are created at compile time in above code?	

Part d. The main function in code asks user to enter the type of transaction and then the amount, what will be the output of code given the user wants to enter following inputs.

```
#include<iostream>
using namespace std;
#include<string>
void getTransType(char &a){
	cout << "Enter W for withdraw and D for deposit" << endl;
	cin >> a;
	if (a != 'W' && a != 'D'){
		string s = "Incorrect Transaction type";
		throw s;
	}
}
void getAmount(int &amount){
	cout << "Enter the amount";
	cin >> amount;

	if (amount<1){
		string s = "Enter positive number";
		throw s;
	}

	else if (amount>5000){
		string s = "Amount should be less than 5001";
		throw s;
	}
}
int main(){
	try{
		char a;
		getTransType(a);
		try{
			int amount;
			getAmount(amount);
			cout << "Successful transaction";
		}
		catch (string e){
			cout << e;
		}
	}
	catch (string ia) {
		cout << ia;
	}
}
```

1) User enters 'W' and then 0 Enter positive number	**2)** User wants to enters 'D' and then 6000 Amount should be less than 5001	**3)** User want to enter 'S' and then 6000 Incorrect Trasaction type