

## PES University, Bengaluru (Established under Karnataka Act No. 16 of 2013)

## JULY 2021: END SEMESTER ASSESSMENT (ESA) B.TECH. VI SEMESTER UE18CS331 - GENERIC PROGRAMMING

Instructions: You may bring any written matter. You cannot have any printed matter. You cannot exchange any material during the exam. Max Marks: 100

Tim	ne: 3 Hr Answer All Questions. Neglect syntax errors Max Marks: 10	00
1. a)	a) answer pricisely. i) Why are template functions defined in a header file and not in an implementation file? ii) How does the compiler and linker take care of multiply instantiated template functions? iii) What is the overhead of template function invocation at runtime? Why? iv) template <typename t1,="" t2="" typename=""> T2 foo(T1); This function requires explicit instantiation. State true or false. Give reasons. v) How does type deduction with the auto keyword work? Is it compile time or runtime mechanism?</typename>	10 [2 + 2 + 2 + 2 + 2 + 2
b) c)	<pre>void what(int a[], int n) {     for(int i = 0; i &lt; n - 1; ++i)         {</pre>	4
	Why? [2 +	2]
. a)	class MyNum {	6
	i) make this a template class.	

why would this code not compile? iii) Support the required operation by providing the required function in the class  [2+2+2]    class MyContainer {			ii) MyNum <int>* x = new MyNum<int>[4];</int></int>		
b) class MyContainer {     public:         MyContainer(int n): n_(n), p_(new int [n_]) {}         -MyContainer() { delete p_; }         private:         int* p_;         int n_; }; int n_; }; i) Fix the bugs in constructor and destructor. Give reasons. ii) Make the class generic.  c) class MyContainer {     private:         int* p_;         int n_; }; The constructor allocates an array of n_ elements and initializes p_ with the pointer to the allocated array. i) write copy constructor ii) Write move assignment operator for this class.  d) Write the swap function to swap two objects of MyContainer using resource stealing using move. Assume that MyContainer supports move assignment and move constructor  a) template <typename ptr="" t=""> ptr_t find_mid(ptr_t first, ptr_t last) {         // TODO } Find an iterator to the middle of a sequence defined by a pair of iterators. Write the code for the following cases i) ptr_t is an input iterator ii) ptr_t is an input iterator iii) ptr_t is an input iterator iiii ptr_t is an input iterator iiii ptr_t is an input iterator iiiii ptr_t is an input iterator iiiii ptr_t is an input iterator iiii ptr_t is an input it</typename>			Why would this code not compile?		
b) class MyContainer {     public:         MyContainer(int n): n_(n), p_(new int [n_]) { }         -MyContainer(int n): n_(n), p_(new int [n_]) { }         -MyContainer(int n): n_(n), p_(new int [n_]) { }         private:         int* p_;         int n_;     };     i) Fix the bugs in constructor and destructor. Give reasons.     ii) Make the class generic.  c) class MyContainer {         private:         int* p_;         int n_;     };     The constructor allocates an array of n_ elements and initializes p_ with the pointer to the allocated array.     ii) write copy constructor     ii) Write move assignment operator for this class.  d) Write the swap function to swap two objects of MyContainer using resource stealing using move. Assume that MyContainer supports move assignment and move constructor  a) template <typename ptr_t="">     ptr_t find_mid(ptr_t first, ptr_t last) {</typename>			iii) Support the required operation by providing the required function in the class		
Comparison of the constructor and destructor. Give reasons.   Comparison of the constructor allocates an array of n_elements and initializes p_with the pointer to the allocated array.   Comparison of the constructor of		1		6	+
public:     MyContainer(nt n): n_(n), p_(new int [n_]) {}     ~MyContainer() { delete p_; }     private:     int* p_;     int n,; }; i) Fix the bugs in constructor and destructor. Give reasons. ii) Make the class generic.  c) class MyContainer {     private:     int* p_;     int n,; }; The constructor allocates an array of n_ elements and initializes p_ with the pointer to the allocated array. i) write copy constructor ii) Write move assignment operator for this class.  d) Wite the swap function to swap two objects of MyContainer using resource stealing using move. Assume that MyContainer supports move assignment and move constructor  a) template <typename -="" a="" an="" and="" apply="" b)="" backwards.="" bidirectional="" by="" can="" cases="" code="" copy="" defined="" do="" each="" element="" find="" find_mid(ptr_t="" first="" first,="" first2,="" first2.<="" following="" for="" from="" fun="" fun)="" fun_b="" fun_t="" i)="" ii)="" input="" is="" iterator="" iterators.="" last)="" last1,="" middle="" modify(ptr_t="" move="" of="" on="" pair="" ptr_2="" ptr_2,="" ptr_b="" ptr_t="" ptr_t,="" requirement:="" second="" sequence="" sequence.="" starting="" td="" template<typename="" the="" through="" to="" todo="" typename="" use="" void="" walk="" which="" write="" {="" }=""><td></td><td>(b)</td><td></td><td>TOTAL ST</td><td></td></typename>		(b)		TOTAL ST	
MyContainer(int n): n, (n), p_(new int [n_]) {}  ~MyContainer() { delete p_; } private: int* p_; int n_; }; i) Fix the bugs in constructor and destructor. Give reasons. ii) Make the class generic.  c) class MyContainer {  private: int* p_; int n; }; The constructor allocates an array of n_ elements and initializes p_ with the pointer to the allocated array. i) write copy constructor ii) Write move assignment operator for this class.  d) Write the swap function to swap two objects of MyContainer using resource stealing using move. Assume that MyContainer supports move assignment and move constructor  a) template <typename ptr_!=""> ptr_t find_mid(ptr_t first, ptr_t last) {</typename>				The second	
-MyContainer() { delete p_; }     private:     int n_;     };     i) Fix the bugs in constructor and destructor. Give reasons.     ii) Make the class generic.  c) class MyContainer {         private:         int* p_;         int n_;     };     The constructor allocates an array of n_ elements and initializes p_ with the pointer to the allocated array.     i) write copy constructor     ii) Write move assignment operator for this class.  d) Write the swap function to swap two objects of MyContainer using resource stealing using move. Assume that MyContainer supports move assignment and move constructor  a) template <typename ptr_t="">     ptr_t find_mid(ptr_t first, ptr_t last)     {</typename>				21	
private:     int* p_;     int n_; }; i) Fix the bugs in constructor and destructor. Give reasons. ii) Make the class generic.  c) class MyContainer {     private:     int* p_;     int n_; }; The constructor allocates an array of n_ elements and initializes p_ with the pointer to the allocated array. i) write copy constructor ii) Write move assignment operator for this class.  d) Write the swap function to swap two objects of MyContainer using resource stealing using move. Assume that MyContainer supports move assignment and move constructor  a) template <typename ptr_t=""> ptr_t find_mid(ptr_t first, ptr_t last) {     // TODO } Find an iterator to the middle of a sequence defined by a pair of iterators. Write the code for the following cases i) ptr_t is an input iterator ii) ptr_t is an input iterator ii) ptr_t is an input iterator ii) ptr_t is an input iterator iii) ptr_t is an input iterator iiii ptr_t is an input iterator iiiit in tr_t in the point in the private in the point in the private in the point in the private in the private in the private in the private in the</typename>					
int n; }; i) Fix the bugs in constructor and destructor. Give reasons. ii) Make the class generic.  c) class MyContainer {     private:         int* p_;         int n_; }; The constructor allocates an array of n_ elements and initializes p_ with the pointer to the allocated array. i) write copy constructor ii) Write move assignment operator for this class.  d) Write the swap function to swap two objects of MyContainer using resource stealing using move. Assume that MyContainer supports move assignment and move constructor  a) template <typename ptr_t="">     ptr_t find_mid(ptr_t first, ptr_t last) {         // TODO } Find an iterator to the middle of a sequence defined by a pair of iterators. Write the code for the following cases i) ptr_t is an input iterator ii) ptr_t is a bidirectional iterator - do use the iterator which can move backwards.  b) template<typename fun_t="" ptr1_t,="" ptr2_t,="" typename=""> void modify(ptr1_t first1, ptr1_t last1, ptr2_t first2, fun_t fun) {         // TODO } Requirement: Walk through the first sequence. apply fun on each element of the first sequence and copy to the second sequence starting from first2.</typename></typename>					
3); i) Fix the bugs in constructor and destructor. Give reasons.   ii) Make the class generic.   4			int* p_;		
i) Fix the bugs in constructor and destructor. Give reasons. ii) Make the class generic.  c) class MyContainer {     private:         int*p;     int n; }; The constructor allocates an array of n_elements and initializes p_with the pointer to the allocated array. i) write copy constructor ii) Write move assignment operator for this class.  d) Write the swap function to swap two objects of MyContainer using resource stealing using move. Assume that MyContainer supports move assignment and move constructor  a) template <typename ptr_t="">     ptr_t find_mid(ptr_t first, ptr_t last) {         // TODO     } Find an iterator to the middle of a sequence defined by a pair of iterators.         Write the code for the following cases i) ptr_t is an input iterator ii) ptr_t is a bidirectional iterator - do use the iterator which can move backwards.  b) template<typename fun_t="" ptr1_t,="" ptr2_t,="" typename="">         void modify(ptr1_t first1, ptr1_t last1, ptr2_t first2, fun_t fun) {         // TODO } Requirement: Walk through the first sequence. apply fun on each element of the first sequence and copy to the second sequence starting from first2.</typename></typename>			int n_;		
ii) Make the class generic.  c) class MyContainer {     private:     int* p.;     int n.; };  The constructor allocates an array of n_ elements and initializes p_ with the pointer to the allocated array. i) write copy constructor ii) Write move assignment operator for this class.  d) Write the swap function to swap two objects of MyContainer using resource stealing using move. Assume that MyContainer supports move assignment and move constructor  a) template <typename -="" a="" an="" b)="" backwards.="" by="" can="" cases="" code="" defined="" do="" find="" find_mid(ptr_t="" first,="" following="" for="" fun_t="" i)="" ii)="" input="" is="" iterator="" iterators.="" last)="" middle="" move="" of="" pair="" ptr1_t,="" ptr2_t,="" ptr_t="" ptr_▷="" sequence="" template<typename="" the="" to="" todo="" typename="" use="" which="" write="" {="" }=""> void modify(ptr1_t first1, ptr1_t last1, ptr2_t first2, fun_t fun) {     // TODO }  Requirement: Walk through the first sequence. apply fun on each element of the first sequence and copy to the second sequence starting from first2.</typename>					
c) class MyContainer {     private:     int* p_;     int n_; };  The constructor allocates an array of n_ elements and initializes p_ with the pointer to the allocated array. i) write copy constructor ii) Write move assignment operator for this class.  d) Write the swap function to swap two objects of MyContainer using resource stealing using move. Assume that MyContainer supports move assignment and move constructor  a) template <typename ptr_t=""> ptr_t find_mid(ptr_t first, ptr_t last) {     // TODO } Find an iterator to the middle of a sequence defined by a pair of iterators. Write the code for the following cases i) ptr_t is an input iterator ii) ptr_t is a bidirectional iterator - do use the iterator which can move backwards.  b) template<typename fun_t="" ptr1_t,="" ptr2_t,="" typename=""> void modify(ptr1_t first1, ptr1_t last1, ptr2_t first2, fun_t fun) {     // TODO } Requirement: Walk through the first sequence. apply fun on each element of the first sequence and copy to the second sequence starting from first2.</typename></typename>			i) Fix the bugs in constructor and destructor. Give reasons.		
Section	-	,			
int* p_;     int n_; }; The constructor allocates an array of n_ elements and initializes p_ with the pointer to the allocated array. i) write copy constructor ii) Write move assignment operator for this class.  d) Write the swap function to swap two objects of MyContainer using resource stealing using move. Assume that MyContainer supports move assignment and move constructor  a) template <typename ptr_t="">     ptr_t find_mid(ptr_t first, ptr_t last) {         // TODO     } Find an iterator to the middle of a sequence defined by a pair of iterators.         Write the code for the following cases i) ptr_t is an input iterator ii) ptr_t is a bidirectional iterator - do use the iterator which can move backwards.  b) template<typename fun_t="" ptr2_t,="" ptrl_t,="" typename="">         void modify(ptrl_t firstl, ptrl_t lastl, ptr2_t first2, fun_t fun)         {</typename></typename>		c)	class MyContainer	4	
int* p_;     int n_; }; The constructor allocates an array of n_ elements and initializes p_ with the pointer to the allocated array. i) write copy constructor ii) Write move assignment operator for this class.  d) Write the swap function to swap two objects of MyContainer using resource stealing using move. Assume that MyContainer supports move assignment and move constructor  a) template <typename ptr_t="">     ptr_t find_mid(ptr_t first, ptr_t last) {         // TODO     } Find an iterator to the middle of a sequence defined by a pair of iterators.         Write the code for the following cases i) ptr_t is an input iterator ii) ptr_t is a bidirectional iterator - do use the iterator which can move backwards.  b) template<typename fun_t="" ptr2_t,="" ptrl_t,="" typename="">         void modify(ptrl_t firstl, ptrl_t lastl, ptr2_t first2, fun_t fun)         {</typename></typename>					
int* p_;     int n_; }; The constructor allocates an array of n_ elements and initializes p_ with the pointer to the allocated array. i) write copy constructor ii) Write move assignment operator for this class.  d) Write the swap function to swap two objects of MyContainer using resource stealing using move. Assume that MyContainer supports move assignment and move constructor  a) template <typename ptr_t="">     ptr_t find_mid(ptr_t first, ptr_t last) {         // TODO     } Find an iterator to the middle of a sequence defined by a pair of iterators.         Write the code for the following cases i) ptr_t is an input iterator ii) ptr_t is a bidirectional iterator - do use the iterator which can move backwards.  b) template<typename fun_t="" ptr2_t,="" ptrl_t,="" typename="">         void modify(ptrl_t firstl, ptrl_t lastl, ptr2_t first2, fun_t fun)         {</typename></typename>					
int n_; }; The constructor allocates an array of n_ elements and initializes p_ with the pointer to the allocated array. i) write copy constructor ii) Write move assignment operator for this class.  d) Write the swap function to swap two objects of MyContainer using resource stealing using move. Assume that MyContainer supports move assignment and move constructor  a) template <typename ptr_t=""> ptr_t find_mid(ptr_t first, ptr_t last) { // TODO } Find an iterator to the middle of a sequence defined by a pair of iterators. Write the code for the following cases i) ptr_t is an input iterator ii) ptr_t is an input iterator ii) ptr_t is a bidirectional iterator - do use the iterator which can move backwards.  b) template<typename fun_t="" ptr2_t,="" ptrl_t,="" typename=""> void modify(ptrl_t first1, ptrl_t last1, ptr2_t first2, fun_t fun) { // TODO } Requirement: Walk through the first sequence apply fun on each element of the first sequence and copy to the second sequence starting from first2.</typename></typename>					
Signature   Sign					
The constructor allocates an array of n_ elements and initializes p_ with the pointer to the allocated array. i) write copy constructor ii) Write move assignment operator for this class.  d) Write the swap function to swap two objects of MyContainer using resource stealing using move. Assume that MyContainer supports move assignment and move constructor  a) template <typename ptr_t=""> ptr_t find_mid(ptr_t first, ptr_t last) {</typename>					
pointer to the allocated array. i) write copy constructor ii) Write move assignment operator for this class.  d) Write the swap function to swap two objects of MyContainer using resource stealing using move. Assume that MyContainer supports move assignment and move constructor  a) template <typename ptr_t=""> ptr_t find_mid(ptr_t first, ptr_t last) {  // TODO } Find an iterator to the middle of a sequence defined by a pair of iterators. Write the code for the following cases i) ptr_t is an input iterator ii) ptr_t is a bidirectional iterator - do use the iterator which can move backwards.  b) template<typename fun_t="" ptr1_t,="" ptr2_t,="" typename=""> void modify(ptr1_t first1, ptr1_t last1, ptr2_t first2, fun_t fun) { // TODO } Requirement: Walk through the first sequence. apply fun on each element of the first sequence and copy to the second sequence starting from first2.</typename></typename>					
i) write copy constructor ii) Write move assignment operator for this class.  d) Write the swap function to swap two objects of MyContainer using resource stealing using move. Assume that MyContainer supports move assignment and move constructor  a) template <typename ptr_t=""> ptr_t find_mid(ptr_t first, ptr_t last) { // TODO } Find an iterator to the middle of a sequence defined by a pair of iterators. Write the code for the following cases i) ptr_t is an input iterator ii) ptr_t is a bidirectional iterator - do use the iterator which can move backwards.  b) template<typename fun_t="" ptr1_t,="" ptr2_t,="" typename=""> void modify(ptr1_t first1, ptr1_t last1, ptr2_t first2, fun_t fun) { // TODO } Requirement: Walk through the first sequence. apply fun on each element of the first sequence and copy to the second sequence starting from first2.</typename></typename>			pointer to the allocated array.		
ii) Write move assignment operator for this class.  d) Write the swap function to swap two objects of MyContainer using resource stealing using move. Assume that MyContainer supports move assignment and move constructor  a) template <typename ptr_t=""> ptr_t find_mid(ptr_t first, ptr_t last) {     // TODO } Find an iterator to the middle of a sequence defined by a pair of iterators. Write the code for the following cases i) ptr_t is an input iterator ii) ptr_t is a bidirectional iterator - do use the iterator which can move backwards.  b) template<typename fun_t="" ptr1_t,="" ptr2_t,="" typename=""> void modify(ptr1_t first1, ptr1_t last1, ptr2_t first2, fun_t fun) {     // TODO } Requirement: Walk through the first sequence. apply fun on each element of the first sequence and copy to the second sequence starting from first2.</typename></typename>					
for this class.     d) Write the swap function to swap two objects of MyContainer using resource stealing using move. Assume that MyContainer supports move assignment and move constructor     a)   template <typename ptr_t=""></typename>					
a) template <typename ptr_t=""> ptr_t find_mid(ptr_t first, ptr_t last) { /// TODO } Find an iterator to the middle of a sequence defined by a pair of iterators. Write the code for the following cases i) ptr_t is an input iterator ii) ptr_t is a bidirectional iterator - do use the iterator which can move backwards.  b) template<typename fun_t="" ptr1_t,="" ptr2_t,="" typename=""> void modify(ptr1_t first1, ptr1_t last1, ptr2_t first2, fun_t fun) { /// TODO } Requirement: Walk through the first sequence. apply fun on each element of the first sequence and copy to the second sequence starting from first2.</typename></typename>			for this class.		
a) template <typename ptr_t=""> ptr_t find_mid(ptr_t first, ptr_t last) { /// TODO } Find an iterator to the middle of a sequence defined by a pair of iterators. Write the code for the following cases i) ptr_t is an input iterator ii) ptr_t is a bidirectional iterator - do use the iterator which can move backwards.  b) template<typename fun_t="" ptr1_t,="" ptr2_t,="" typename=""> void modify(ptr1_t first1, ptr1_t last1, ptr2_t first2, fun_t fun) { /// TODO } Requirement: Walk through the first sequence. apply fun on each element of the first sequence and copy to the second sequence starting from first2.</typename></typename>		d)	Write the swap function to swap two objects of MyContainer using resource stealing	4	
ptr_t find_mid(ptr_t first, ptr_t last)  {  // TODO } Find an iterator to the middle of a sequence defined by a pair of iterators.  Write the code for the following cases i) ptr_t is an input iterator ii) ptr_t is a bidirectional iterator - do use the iterator which can move backwards.  b) template <typename fun_t="" ptr1_t,="" ptr2_t,="" typename=""> void modify(ptr1_t first1, ptr1_t last1, ptr2_t first2, fun_t fun)  {  // TODO } Requirement:  Walk through the first sequence. apply fun on each element of the first sequence and copy to the second sequence starting from first2.</typename>			using move. Assume that MyContainer supports move assignment and move constructor		
ptr_t find_mid(ptr_t first, ptr_t last)  {  // TODO } Find an iterator to the middle of a sequence defined by a pair of iterators.  Write the code for the following cases i) ptr_t is an input iterator ii) ptr_t is a bidirectional iterator - do use the iterator which can move backwards.  b) template <typename fun_t="" ptr1_t,="" ptr2_t,="" typename=""> void modify(ptr1_t first1, ptr1_t last1, ptr2_t first2, fun_t fun)  {  // TODO } Requirement:  Walk through the first sequence. apply fun on each element of the first sequence and copy to the second sequence starting from first2.</typename>	_				
ptr_t find_mid(ptr_t first, ptr_t last)   {	1	a)		18	
A   Find an iterator to the middle of a sequence defined by a pair of iterators.   Write the code for the following cases   i) ptr_t is an input iterator   ii) ptr_t is a bidirectional iterator - do use the iterator which can move backwards.   b)   template <typename fun_t="" ptr2_t,="" ptrl_t,="" typename="">   void modify(ptrl_t first1, ptr1_t last1, ptr2_t first2, fun_t fun)   8   [4  </typename>			ptr_t find_mid(ptr_t first, ptr_t last)		
Find an iterator to the middle of a sequence defined by a pair of iterators.  Write the code for the following cases i) ptr_t is an input iterator ii) ptr_t is a bidirectional iterator - do use the iterator which can move backwards.  b) template <typename fun_t="" ptr2_t,="" ptrl_t,="" typename=""> void modify(ptrl_t first1, ptrl_t last1, ptr2_t first2, fun_t fun)  { // TODO }  Requirement: Walk through the first sequence. apply fun on each element of the first sequence and copy to the second sequence starting from first2.</typename>			{	+	
Find an iterator to the middle of a sequence defined by a pair of iterators.  Write the code for the following cases i) ptr_t is an input iterator ii) ptr_t is a bidirectional iterator - do use the iterator which can move backwards.  b) template <typename fun_t="" ptr2_t,="" ptrl_t,="" typename=""> void modify(ptrl_t first1, ptrl_t last1, ptr2_t first2, fun_t fun)  { /// TODO }  Requirement: Walk through the first sequence. apply fun on each element of the first sequence and copy to the second sequence starting from first2.</typename>			// TODO	41	
Write the code for the following cases  i) ptr_t is an input iterator  ii) ptr_t is a bidirectional iterator - do use the iterator which can move backwards.  b) template <typename fun_t="" ptr1_t,="" ptr2_t,="" typename=""> void modify(ptr1_t first1, ptr1_t last1, ptr2_t first2, fun_t fun)  { /// TODO }  Requirement: Walk through the first sequence. apply fun on each element of the first sequence and copy to the second sequence starting from first2.</typename>				,	
i) ptr_t is an input iterator ii) ptr_t is a bidirectional iterator - do use the iterator which can move backwards.  b) template <typename fun_t="" ptr1_t,="" ptr2_t,="" typename=""> void modify(ptr1_t first1, ptr1_t last1, ptr2_t first2, fun_t fun)  { /// TODO }  Requirement: Walk through the first sequence. apply fun on each element of the first sequence and copy to the second sequence starting from first2.</typename>			Find an iterator to the middle of a sequence defined by a pair of iterators.		
ii) ptr_t is a bidirectional iterator - do use the iterator which can move backwards.  b) template <typename fun_t="" ptr1_t,="" ptr2_t,="" typename=""> void modify(ptr1_t first1, ptr1_t last1, ptr2_t first2, fun_t fun)  { /// TODO }  Requirement: Walk through the first sequence. apply fun on each element of the first sequence and copy to the second sequence starting from first2.</typename>					
b) template <typename fun_t="" ptr2_t,="" ptrl_t,="" typename=""> void modify(ptrl_t first1, ptrl_t last1, ptr2_t first2, fun_t fun)  {  /// TODO }  Requirement: Walk through the first sequence. apply fun on each element of the first sequence and copy to the second sequence starting from first2.</typename>					
b) template <typename fun_t="" ptr1_t,="" ptr2_t,="" typename=""> void modify(ptr1_t first1, ptr1_t last1, ptr2_t first2, fun_t fun)  { /// TODO }  Requirement: Walk through the first sequence. apply fun on each element of the first sequence and copy to the second sequence starting from first2.</typename>					
void modify(ptr1_t first1, ptr1_t last1, ptr2_t first2, fun_t fun)  {  /// TODO }  Requirement: Walk through the first sequence. apply fun on each element of the first sequence and copy to the second sequence starting from first2.	L	1			
Requirement: Walk through the first sequence. apply fun on each element of the first sequence and copy to the second sequence starting from first2.	U	/	void modify(ntr1 t first1 ntr1 t lost1 ntr2 t first2 first2 first3 first	8	
Requirement: Walk through the first sequence. apply fun on each element of the first sequence and copy to the second sequence starting from first2.			void modify(pur_t mstr, pur_t lastr, pur2_t mst2, fun_t fun)		
Requirement: Walk through the first sequence. apply fun on each element of the first sequence and copy to the second sequence starting from first2.			// TODO		
Requirement: Walk through the first sequence. apply fun on each element of the first sequence and copy to the second sequence starting from first2.			}	2	
Walk through the first sequence. apply fun on each element of the first sequence and copy to the second sequence starting from first2.					
Walk through the first sequence. apply fun on each element of the first sequence and copy to the second sequence starting from first2.			Requirement:	2	27
and copy to the second sequence starting from first2.			Walk through the first sequence, apply for an arely 1		7
i) complete the code			and copy to the second sequence starting from first sequence		
			i) complete the code		

		ii) how many arguments can fun take? why?	
-		iii) Why the second sequence is identified by a single iterator?	
	c)	Develop a generic predicate functor class to check whether the given value is between a pair of given values.	4
		Ex: Bet(10, 20)(15) should be true as 15 is in between 10 and 20.	
			1.0
4	a)	i) what does pop_back on vector return? why?	10
		ii) Why we cannot implement queue using a vector?	
		iii) What all gets invalidated on change in size of vector? Why?	
		iv) What is the difference between [] and at member functions on vector	
	1	v) Why set does not provide indexing operator?	5
	b)	find the number of unique elements in a given sequence. You cannot modify the	3
		given sequence. You can assume that < is supported on the component type. template <typename ptr="" t=""></typename>	
		Hint: You may have to find the component type	
		int count_uniq(ptr_t first, ptr_t last)	
		{	
		// ToDo	
		}	
	c)	struct Term	5
		{	
		int coeff_;	
		int expo_;	
		}; !:	
		List <term> poly;</term>	
		Poly represents a polynomial stored in a list form with each node having coefficient and exponent.	
		Given a polynomial, differentiate the polynomial.	
		void diff(const List <term>&amp; src, List<term>&amp; dst)</term></term>	
		{	
		// TODO	
		}	
5	a)	can we support the following in java? Give reasons.	6
		i) class A <t> { } A<int> x = new A<int>;</int></int></t>	
		ii) class A <t> {</t>	
		static int x;	
		}	
		iii) class A <t> { }</t>	
		A <double> a1 = new A<double>;</double></double>	
		A <double> a2 = new A<double>;</double></double>	
		A < Double > a3 = a1 + a2; [6]	
			1 200
5	b)	How are the following supported in C#?	4
		i) operator[]	
		ii) read only property	
		give a simple example	
5	c)	Write template metaprogram to find a to the power m.	4

		Assume $m \ge 0$	
5	d)	This is the definition of unordered map.	6[
		template < class Key,	2+
		class T,	2+
		class Hash =,	2]
		class Pred =,	
		class Alloc =	
		> class unordered_map;	
		i) what is the significance of the type Hash?	
		ii) Why do we require Pred?	
		iii) What would each element of unordered map contain?	
1	1		