Derek Rodriguez, Derek Caprio COP 4530, Project 2 Report and UML Diagrams

## Report:

This project implements a stack and a single-ended queue data structure using dynamic arrays. The menu program contains an insert option which calls enqueue or push for queue or stack respectively and a delete option which calls dequeue or pop for queue or stack respectively as well as other common menu items such as display and clear in addition to options to check the current size of the dynamic array and the number of elements it currently stores.

Both data structures use a dynamic array for their implementations, They also both resize in the same way: if an item is inserted into an array that is full, the array size is doubled; and if after deleting an item, the array is ½ full, then the array size is halved. In order to resize the data structure, a new array is built that is of the required size, then each element is copied into the new array, after this, the old array is deleted. Due to this resizing, the insertion and delete (push, enqueue, pop, and dequeue) were the most involved. Functions such as empty, size, and capacity were easily implemented with a couple lines of code.

Our group method of wording on this project was that Derek R. wrote the bulk of the queue header file and Derek C. worked on the bulk of the stack header file. After this, debugging and testing was shared equally with both of us working on both classes. We spent between 5 and 10 hours each on the project.

[UML diagram on next page]

UML Diagrams:

	PROJECT 2 UML DIAGRAMS
Dy.	Stack
1-1	T* GITSY
	Int count
	int initialSize
1-	10t arraySize
+	Pynstack (int n = 15) // construttor
1+	Dynstrick (const Dynstack & Stack) // copy constructor
1	~ Dynstack () // destruitor
1+	T top (void) const
1	int size (void) const
1+	bool empty (word) const
1+	int capacity (vaid) const
1	void display (void)
+	void push (+ const & data)
	T 70p (voja)
+ +	Dynsterk & operator = (const. Dynsterk& Sycar) Moverlanded assignment
Di	a Que
-	Type X array
-	Type X array
-	
-	Int literal Int iTail Int count
1111	Int I Head  Int i Tail  Int count  Int initial 5 13c
	Int literal Int iTail Int count Int initials ite
1	Int I Head  Int i Tail  Int count  Int Initials ite  Int arraySize  Dynaue (int n = 15) // Constructor
	Int I Head  Int i Tail  Int count  Int Initials ite  Int arraySize  Dynaue (Int n = 15) // Constructor  No Dynaue () // destructor
++	Int itail Int count Int instals like Int arraySize  Dynaue (Int n = 15) // Constructor  or Dynaue () // destructor  type front (vaid) const
	Int itall Int count Int initials ite Int arraySize  DynGue (int n = 15) // Constructor  rype front (vaid) const  Type back (vaid) const  Type back (vaid) const
+	Int itail Int count Int instals like Int arraySize  Dynaue (Int n = 15) // Constructor  represent (vaid) const  Type front (vaid) const  Type back (void) const  Int Size (void) const
++	Int itall Int count Int instals ite Int arraySize  DynGue (Int n = 15) // constructor  re DynGue () // destructor  Type front (vaid) const  Type back (void) const  Int size (void) const  bool empty(void) const  bool empty(void) const
+++	Int ited  Int ital!  Int count  Int initials ite  Int arraySize  DynSive (int n = 15) // Constructor  A DynSive () // Jestructor  Type front (vaid) const  Type back (void) const  Int size (void) const  back empty(vaid) const  back empty(vaid) const  Int (cyacity (void) const  Int (cyacity (void) const
+++	Int itself  Int count  Int initials lite  Int arraysize  Dyngue (int n = 15) // constructor  or Dyngue ()  Type front (vaid) const  Type back (void) const  Int size (void) const  bool empty (void) const  Int (size(void) const  Void display (void)  Void display (void)
++++	Int ited  Int ited  Int count  Int instals ite  Int arraysite  Dynave (int n = 15) // constructor  or Dynave () // destructor  Type front (vaid) const  Type back (void) const  Int size (void) const  bool empty (void) const  Int capacity (void) const  void display (void)  void enquel Type const & data)
+++++	Int itself  Int count  Int initials lite  Int arraysize  Dyngue (int n = 15) // constructor  or Dyngue ()  Type front (vaid) const  Type back (void) const  Int size (void) const  bool empty (void) const  Int (size(void) const  Void display (void)  Void display (void)