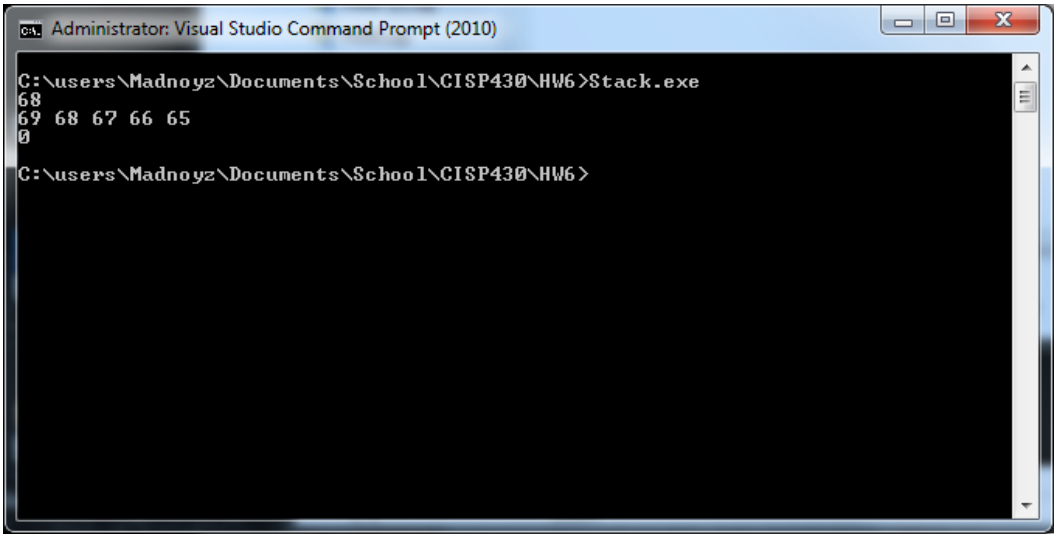
```
/*****************************************************************************************************
**
**      Name: Clyde Pabro
**      Class:          CISP430 - Fall 2012 Thu
**      Assignment:   Homework 6
**
*****************************************************************************************************/
```

/***
Program 1: Stack using Linked Nodes.
***/

OUTPUT:



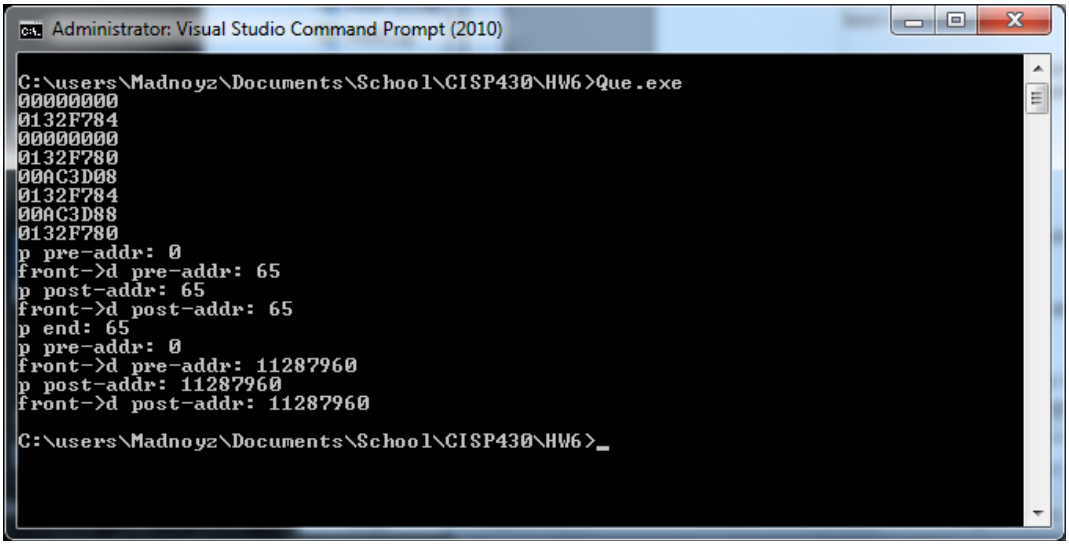
SOURCE CODE:

```
#include <iostream>
#include <cstdlib>
using namespace std;

typedef int data;

struct node{
	data d;
	struct node *next;
};

node *head = 0;
node *tail = 0;

void push ( data );
data pop( void );
data peek( void );
bool isEmpty( void );

int main( void )
{
	push('A');
	push('B');
	push('C');
	push('D');
	push('E');
	push('F');
	pop();
	cout << peek() << endl;
	while( !isEmpty() )
```

```
	{
		cout << pop() << " ";
	}
	cout << "\n" << peek() << endl;
	return 0;
}

void push ( data d )
{
	node *temp = (node*)malloc(sizeof(node));
	temp->d = d;
	temp->next = head;
	head = temp;
}

data pop( void )
{
	data d;
	node *temp;
	if( isEmpty() )
		return 0;
	temp = head;
	head = head->next;
	d = temp->d;
	free (temp);
	return d;
}

data peek( void )
{
	if( isEmpty() )
		return 0;

	return head->next->d;
}

bool isEmpty( void )
{
	if( head )
		return false;
	else
		return true;

}
```

/***
Program 2: Queue using Linked Nodes.
***/

OUTPUT:



SOURCE CODE:

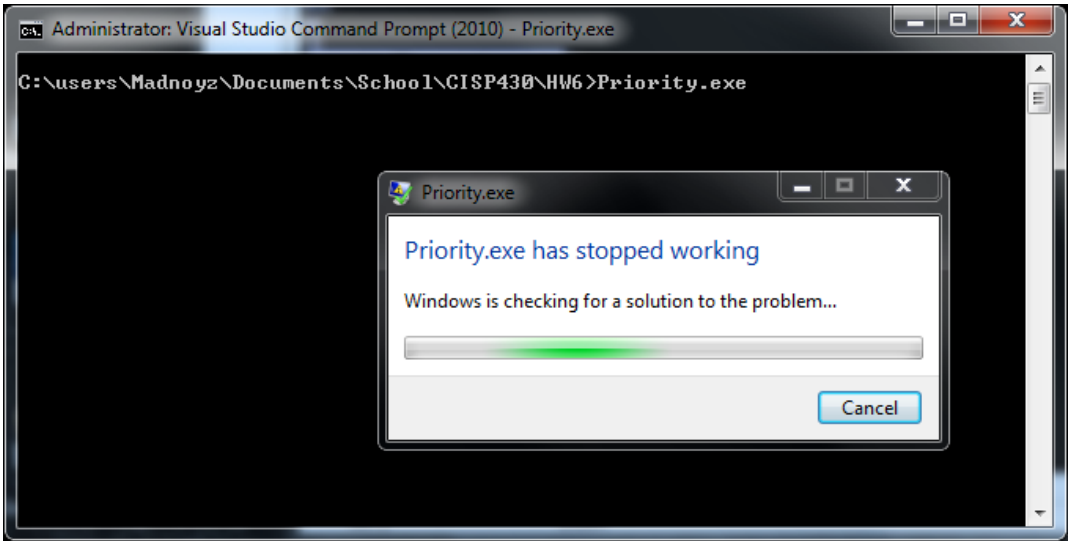
```
#include <iostream>
#include <cstdlib>
using namespace std;

typedef int data;

struct node {
	data d;
	struct node *next;
};

struct node *front = 0;
struct node *rear;

void q( data );
data dq( void );
bool isEmpty( void );

int main()
{
	cout << front << endl;
	cout << &front << endl;
	cout << rear << endl;
	cout << &rear << endl;
	q('A');
	q('B');
	q('C');
	q('D');
	q('E');
	q('F');
```

```
	cout << front << endl;
	cout << &front << endl;
	cout << rear << endl;
	cout << &rear << endl;
	dq();
	while( !isEmpty() )
	{
		dq();
	}
	cout << "End front " << front << endl;
	cout << "End front addr " << &front << endl;
	cout << "End rear " << rear << endl;
	cout << "End rear addr " << &rear << endl;
	return 0;
}

void q( data d )
{
	node *newNode = (node*)malloc(sizeof(node));
	newNode->d = d;
	newNode->next = NULL;
	rear = newNode;
	rear->next = newNode;
	if ( front == 0 )
		front = rear;
}

data dq( void )
{
	data p = 0;
	struct node *temp;
	if( isEmpty() )
	{
		cout << "\nQueue is empty!" << endl;
		return 0;
	} else {
		temp = front;
		cout << "p pre-addr: " << p << endl;
		cout << "front->d pre-addr: " << front->d << endl;
		p = front->d;
		cout << "p post-addr: " << p << endl;
		cout << "front->d post-addr: " << front->d << endl;
		front = front->next;
		free (temp);
	}
	cout << "p end: " << p << endl;
	return p;
}

bool isEmpty( void )
{
```

```
        if( front )
        {
                return false;
        }else{

                return true;
        }
}
```

/**
Program 3: Priority Queue using Linked Nodes.
**/

OUTPUT:



Segmentation Fault at insert()

SOURCE CODE:

```
#include <iostream>
#include <cstdlib>
using namespace std;

typedef int data;

struct node {
	data d;
	struct node *next;
};

struct node *head = NULL;
struct node *tail = NULL;

void insert( data );
data dq( void );
data peek( void );
bool isEmpty( void );

int main()
{
	insert('A');
	insert('B');
	insert('C');
	insert('D');
	insert('E');
	insert('F');
	dq();
	cout << peek() << endl;
```

```
	while( !isEmpty() )
	{
		cout << dq() << " ";
	}
	cout << "\n" << peek() << endl;
	return 0;
}

void insert( data d )
{
	data count;
	node *c;
	node *pc;
	struct node *p = (node*)malloc(sizeof(node));

	if( !p )
		return;

	p->d = d;
	p = head;

	if ( head )
	{
		p->next = c;
		head = p;
	} else {

		while(( c != NULL ) && ( count < d - 1 ))
		{
			count++;
			pc = c;
			c = c->next;
		}

		if( p == NULL )
		{
			pc->next = p;
			p->next = NULL;
		} else {
			pc->next= p ;
			p->next = c;
		}
	}

}

data dq( void )
{
	data p = 0;
	struct node *temp;
	if( isEmpty() )
```

```
	{
		cout << "\nQueue is empty!" << endl;
		return 0;
	} else {
		temp = head;
		cout << "p pre-addr: " << p << endl;
		cout << "head->d pre-addr: " << head->d << endl;
		p = head->d;
		cout << "p post-addr: " << p << endl;
		cout << "head->d post-addr: " << head->d << endl;
		head = head->next;
		free (temp);
	}
	cout << "p end: " << p << endl;
	return p;
}

data peek( void )
{
	if( isEmpty() )
		return 0;

	return head->next->d;
}

bool isEmpty( void )
{
	if( head )
		return false;
	else
		return true;
}
```