Current State		Input	Next State	Output
Mode	Calculator	Button Press		(Change)
INFIX	Nothing/Start	key(opr), key (equ)	Nothing/Start	N/A
INFIX	Nothing/Start	key(num)	Build Number	n_i = key(num)
INFIX	Build Number	key(num)	Build Number	n_i = concate(n_i, key(num))
INFIX	Build Number	key(opr)	Add Operator	ni; opr= key(opr)
INFIX	Build Number	key(equ)	Compute	ans = ans + apply(opr, ni…)
INFIX	Build Number	key(dec)	Decimal	ni = concate(ni, key(dec))
INFIX	Decimal	key(dec)	Decimal	ni = trunc(ni)
INFIX	Decimal	key(equ), key (opr)	Decimal	ni
INFIX	Decimal	key(num)	Build Number	n_i = concate(n_i, key(num))
INFIX	Add Operator	key(opr)	Add Operator	ni; opr= key(opr)
INFIX	Add Operator	key(equ)	Add Operator	N/A
INFIX	Add Operator	key(num)	Build Number	N/A
INFIX	Compute	key(num)	Build Number	ni= key(num)
INFIX	Compute	key(opr)	Add Operator	ans; opr=key(opr)
INFIX	Compute	key(num)	Build Number	n_i = key(num)
INFIX	Nothing/Start	key(postfix)	POSTFIX Nothing/Start	0
INFIX	Build Number	key(postfix)	POSTFIX Nothing/Start	0
INFIX	Decimal	key(postfix)	POSTFIX Nothing/Start	0
INFIX	Add Operator	key(postfix)	POSTFIX Nothing/Start	0
INFIX	Compute	key(postfix)	POSTFIX Nothing/Start	0
POSTFIX	Nothing/Start	key(opr), key (equ)	Nothing/Start	N/A
POSTFIX	Nothing/Start	key(num)	Build Number	n_i = key(num)
POSTFIX	Build Number	key(num)	Build Number	n_i = concate(n_i, key(num))
POSTFIX	Build Number	key(equ)	Build Number	N/A
POSTFIX	Build Number	key(enter)	Switch	ni
POSTFIX	Build Number	key(dec)	Decimal	ni = concate(ni, key(dec))
POSTFIX	Decimal	key(dec)	Decimal	ni = trunc(ni)
POSTFIX	Decimal	key(equ), key (opr)	Decimal	ni
POSTFIX	Decimal	key(num)	Build Number	n_i = concate(n_i, key(num))
POSTFIX	Switch	key(num)	Build Number	n(++i) = key(num)
POSTFIX	Switch	key(opr)	Compute	ans = n(--i) = apply (n(--i), key(opr), ni)
POSTFIX	Compute	key(opr)	Compute	ans = n(--i) = apply (n(--i), key(opr), ni)
POSTFIX	Compute	key(num)	Build Number	ans; i--
POSTFIX	Nothing/Start	key(infix)	INFIX Nothing/Start	0
POSTFIX	Build Number	key(infix)	INFIX Nothing/Start	0
POSTFIX	Decimal	key(infix)	INFIX Nothing/Start	0
POSTFIX	Switch	key(infix)	INFIX Nothing/Start	0
POSTFIX	Compute	key(infix)	INFIX Nothing/Start	0

```
#include<iostream>
#include<stdio.h>
#include<ctype.h>
#include<math.h>
using namespace std;

class calc {
	public:
		double n[100] = { 0.0 },ans = 0.0;
		int dec_pow = -1;
		int i = 0;
		char opr[99] = { 0 };
		bool dec= false, fix = true;

		void clear() {
			n[100] = { 0.0 };
			i = 0;
			opr[99] = { 0 };
			ans = 0.0;
			dec = false;
			dec_pow = -1;
		}
};

//State 001 NOTHING/START
void startINFIX(calc a, char key);
//State 002 BUILD NUMBER
void buildNumber(calc a, char key);
void addDec(calc a, char key);
//State 003 INSERT OPR
void insertOpr(calc a, char key);
//State 004 COMPUTE
void compute(calc a, char key);
double apply(calc a);

//State 010 NOTHING
void startPFIX(calc a, char key);
//State 020 BUILD NUMBER
void buildNumberP(calc a, char key);
//State 030 INSERT OPR
void insertOprP(calc a, char key);
//State 040 COMPUTE
void computeP(calc a, char key);


//SUPPORT FUNCTIONS
int buttonReader(char key);
bool isOpr(char key);
int toInt(char key);

int main() {
	char key = 0;
	calc a;
	startINFIX(a, key);
	cin.ignore();
}

void startINFIX(calc a, char key) {
	int input = 0;
	a.clear();
	a.fix = true;
	std::cout << "\n INFIX Calculator Has Been
Cleared. Ready To Begin \n";
	std::cin >> key;

	input = buttonReader(key);
	if (input == 1)
		buildNumber(a, key);
	else if (input == 6)
		startPFIX(a,key);
	else
		startINFIX(a, key);
}
void startPFIX(calc a, char key) {
	int input = 0;
	a.fix = false;
	a.clear();
	std::cout << "\n PFIX Calculator Has Been
Cleared. Ready To Begin \n";
	std::cin >> key;

	input = buttonReader(key);
	if (input == 1)
		buildNumberP(a, key);
	else if (input == 6)
		startINFIX(a,key);
	else
		startPFIX(a, key);
}


void buildNumber(calc a, char key) {
	if (a.dec == false) {
		if (a.n[a.i] == 0)
			a.n[a.i] = toInt(key);
		else
			a.n[a.i] = (a.n[a.i]* 10) +
toInt(key);
		}
	else {
			a.n[a.i] = a.n[a.i] + (pow(10,
a.dec_pow)*toInt(key));
			a.dec_pow--;
		}
	std::cout << '[' << a.i << "]  " << a.n[a.i];

	std::cout << "\n \n Continue to build number";
	std::cout << a.i;
	std::cout << " or type opr to continue \n";
	std::cout << "IN: ";
	cin >> key;
	int x = buttonReader(key);
	std::cout << "X " << x;
	if (x == 1) {
		buildNumber(a, key);
	}
	else if (x == 2) {
		insertOpr(a,key);
	}
	else if (x == 3) {
		compute(a, key);
	}
	else if (x == 4) {
		addDec(a, key);
	}
	else if (x == 6) {
		startPFIX(a, key);
	}
	else
		std::cout << "ERROR a button has been
pressed that is not actualy on the calculator";
}
void addDec(calc a, char key) {
	a.dec = true;
	std::cout << '[' << a.i << "]  " << a.n[a.i]
<<'.';
	std::cout << "\n \n Continue to build number";
	std::cout << a.i;
	std::cout << " or type opr to continue \n";
	std::cout << "IN: ";
	cin >> key;
	if (buttonReader(key) == 1) {
		buildNumber(a, key);
	}
	else
		addDec(a, key);
}

void buildNumberP(calc a, char key) {
	if (a.dec == false) {
		if (a.n[a.i] == 0)
			a.n[a.i] = toInt(key);
		else
			a.n[a.i] = (a.n[a.i] * 10) +
toInt(key);
		}
	else {
```

```
                    a.n[a.i] = a.n[a.i] + (pow(10,
a.dec_pow)*toInt(key));
                    a.dec_pow--;
          }
          std::cout << '[' << a.i << "]  " << a.n[a.i];

          std::cout << "\n \n Continue to build number";
          std::cout << a.i;
          std::cout << " or type opr to continue \n";
          std::cout << "IN: ";
          cin >> key;
          int x = buttonReader(key);
          std::cout << "X " << x;
          if (x == 1) {
                    buildNumberP(a, key);
          }
          else if (x == 2) {
                    insertOprP(a, key);
          }
          else if (x == 4) {
                    addDec(a, key);
          }
          else if (x == 5) {
                    a.i++;
                    buildNumberP(a, key);;
          }
          else if (x == 6) {
                    startINFIX(a, key);
          }
          else
                    std::cout << "ERROR a button has been
pressed that is not actualy on the calculator";
}

void insertOpr(calc a, char key) {
          a.opr[a.i] = key;
          std::cout << a.opr[a.i]  << '[' << a.i << "]
" << a.n[a.i];
          std::cout << "\n \n OPR ADDED to ";
          std::cout << a.i;
          std::cout << " or type opr to change \n";
          std::cout << "IN: ";
          cin >> key;
          int x = buttonReader(key);
          if (x == 1) {
                    a.i++;
                    a.dec_pow = -1;
                    a.dec = false;
                    buildNumber(a, key);
          }
          else if (x == 2) {
                    insertOpr(a, key);
          }
          else if (x == 6) {
                    //Switch
          }
          else
                    std::cout << "ERROR a button has been
pressed that is not actualy on the calculator";
}
void insertOprP(calc a, char key) {
          a.opr[a.i] = key;
          std::cout << a.opr[a.i] << '[' << a.i << "]  "
<< a.n[a.i];
          std::cout << "\n \n OPR ADDED to ";
          std::cout << a.i;

          computeP(a, key);
}


void compute(calc a, char key) {
          a.ans= apply(a);
          std::cout << "\n Result: ";
          std::cout << a.ans;
          std::cout << "\n";

          std::cout << "\n \n Continue to build number";
          std::cout << a.i;
          std::cout << " or type opr to continue \n";
          std::cout << "IN: ";
          cin >> key;
          int x = buttonReader(key);
          std::cout << "X " << x;
          if (x == 1) {
                    a.i = 1;
                    a.n[0] = a.ans;
                    buildNumber(a, key);
          }
          else if (x == 6) {
                    //Switch
          }
          else
                    std::cout << "ERROR a button has been
pressed that is not actualy on the calculator";
}
void computeP(calc a, char key) {
          if (a.opr[a.i] == '+') {
                    a.n[a.i - 1] = a.n[a.i - 1] +
a.n[a.i];
          }
          else if (a.opr[a.i] == '-') {
                    a.n[a.i - 1] = a.n[a.i - 1] -
a.n[a.i];
          }
          else if (a.opr[a.i] == '*') {
                    a.n[a.i - 1] = a.n[a.i - 1] *
a.n[a.i];
          }
          else if (a.opr[a.i] == '/') {
                    a.n[a.i - 1] = a.n[a.i - 1] /
a.n[a.i];
          }
          else
                    cout << "ERROR \n";

                    a.i--;


          std::cout << " or type opr to continue \n";
          std::cout << "IN: ";
          cin >> key;
          int x = buttonReader(key);
          if (x == 1) {
                    buildNumberP(a, key);
          }
          else if (x == 2) {
                    insertOpr(a, key);
          }
          else
                    startPFIX(a, key);
}

double apply(calc a) {
          double temp =0;
          int i = a.i;
          for (int x = 0; x < i; x++) {
                    if (a.opr[x] == '*') {
                              a.n[x] = a.n[x] * a.n[x+1];
                              for (int j = x + 1; j <= i;
j++)
                                        a.n[j] = a.n[j+1];
                              for (int j = x + 1; j <= i;
j++)
                                        a.opr[j] =
a.opr[j+1];

                              i--;
                    }
                    else if (a.opr[x] == '/') {
                              a.n[x] = a.n[x] / a.n[x +
1];
                              for (int j = x + 1; j <= i;
j++)
                                        a.n[j] = a.n[j +
1];
                              for (int j = x + 1; j <= i;
j++)
```

```
                                        a.opr[j] = a.opr[j
+ 1];

                                i--;
                        }
                        else {

                        }
                }
                if (i < 1)
                        i = 2;
                for (int x = 0; x < i; x++) {
                        if (a.opr[x] == '+') {
                                temp = a.n[x] + a.n[x + 1];
                        }
                        else{
                                temp = a.n[x] - a.n[x + 1];
                        }
                }
                return temp;
}

int buttonReader(char key) {
        if (isdigit(key)) {
                return (1);
        }
        else if (isOpr(key)) {
                return (2);
        }
        else if (key == '=') {
                return (3);
        }
        else if (key == '.') {
                return (4);
        }
        else if (key == 'n') {
                return(5);
        }
        else if (key == 'T') {
                return (6);
        }
        else {
                std::cout << "ERROR \n";
                return(0);
        }
}
bool isOpr(char key) {
        if (key == '+') {
                return true;
        }
        else if (key == '-')
                return true;
        else if (key == '*')
                return true;
        else if (key == '/')
                return true;
        else
                return false;
}
int toInt(char key) {
        if (key == '0') {
                return 0;
        }
        else if (key == '1') {
                return 1;
        }
        else if (key == '2') {
                return 2;
        }
        else if (key == '3') {
                return 3;
        }
        else if (key == '4') {
                return 4;
        }
        else if (key == '5') {
                return 5;
        }
        else if (key == '6') {
                return 6;
        }
        else if (key == '7') {
                return 7;
        }
        else if (key == '8') {
                return 8;
        }
        else
                return 9;
}
```