Q1:

dayType.h

#ifndef P1\_DAYTYPE\_H  
#define P1\_DAYTYPE\_H  
  
#include <string>  
  
using namespace std;  
  
class dayType  
{  
public:  
 static string weekDays[7];  
 // Function to build array in the form of weekDays[7] // containing the days of the week  
  
 void print() const;  
 // Function to output weekday in the form of weekDay  
  
 string nextDay() const;  
 // Function to determine the next day based on the value // in weekDay  
 // Postcondition: The next day value is returned  
  
 string prevDay() const;  
 // Function to determine the previous day based on the // value in weekDay  
 // Postcondition: The previous day value is returned  
  
 void addDay(int nDays);  
 // Function to determine a day in the future according to // the parameter  
 // Postcondition: The future day value is returned  
  
 void setDay(string d);  
 // Function to set the day according to the parameter  
 // Postcondition: weekDay = d  
  
 string getDay() const;  
 // Function to return the week day  
 // Postcondition: The value of weekDay is returned  
  
 dayType();  
 // Default constructor  
 // Sets weekDay to a null string  
 // Postcondition: weekDay = ""  
  
 dayType(string d);  
 // Constructor with parameters  
 // Sets weekDay according to the parameter  
 // Postcondition: weekDay = d  
  
private:  
 string weekDay; // Variable to store week day  
};  
  
#endif //P1\_DAYTYPE\_H

dayType.cpp

#include "dayType.h"  
#include <iostream>  
#include <string>  
using namespace std;  
  
string dayType::weekDays[7] = {"Sunday", "Monday", "Tuesday",  
 "Wednesday", "Thursday", "Friday",  
 "Saturday"};  
  
void dayType::print() const  
{  
 cout << weekDay;  
}  
  
string dayType::nextDay() const  
{  
 int i;  
  
 for (i = 0; i < 7; i++)  
 if (weekDays[i] == weekDay)  
 break;  
 return weekDays[(i + 1) % 7];  
}  
  
string dayType::prevDay() const  
{  
 if (weekDay == "Sunday")  
 return "Saturday";  
 else  
 {  
 int i;  
  
 for (i = 0; i < 7; i++)  
 if (weekDays[i] == weekDay)  
 break;  
 return weekDays[i - 1];  
 }  
}  
  
void dayType::addDay(int nDays)  
{  
 int i;  
  
 for (i = 0; i < 7; i++)  
 if (weekDays[i] == weekDay)  
 break;  
 weekDay = weekDays[(i + nDays) % 7];  
}  
  
void dayType::setDay(string d)  
{  
 weekDay = d;  
}  
  
string dayType::getDay() const  
{  
 return weekDay;  
}  
  
dayType::dayType()  
{  
 weekDay = "Sunday";  
}  
  
dayType::dayType(string d)  
{  
 weekDay = d;  
}

Main.cpp

#include <iostream>  
#include "dayType.h"  
using namespace std;  
  
int main()  
{  
  
 dayType myDay;  
  
 //declare variables  
 string weekday;  
 int daysAhead;  
  
 //get current day from user , call setDay function and pass weekday  
 cout << "Current day - ";  
 cin >> weekday;  
 myDay.setDay(weekday);  
  
 //call prevDay Function to display next day  
 cout << "Previous day - ";  
 cout << myDay.prevDay();  
 cout << endl;  
  
 //call nextDay function to display next day  
 cout << "Next day - ";  
 cout << myDay.nextDay();  
 cout << endl << endl;  
  
 //collect days ahead from user, call addDay function and pass days ahead to add the days  
 cout << "Enter days ahead: ";  
 cin >> daysAhead;  
 myDay.addDay(daysAhead);  
  
 // prompts user of days and displays the day for the week for those number of days  
 cout << daysAhead << " days away - ";  
 cout << myDay.getDay();  
 cout << endl << endl << "End of Program";  
  
 return 0;  
}

Output:

Current day - Sunday

Previous day - Saturday

Next day - Monday

Enter days ahead: 4

4 days away - Thursday

End of Program

Q2:

1. Pseudo Code of Stack

**Insert an element in stack**

Insertion (a, size)

full = size – 1

If full then

print ‘STACK OVERFLOW’

exit

Else

Top = 0 // stack empty

top = top+1

end if

a[top]=item

Exit

**Delete an element from stack**

Deletion(item)

If top = 0 then

print ’STACK Empty’

exit

Else

Delete item

end if

top = top -1

Exit

**Display element of stack**

Display()

If top = 0 then

Print ‘STACK EMPTY’

Exit

Else

For i=top to 0

Print a[i]

End for

Exit

1. Pseudo Code of Queue

**Insert an element in queue**

Insert rear()

[check for queue is over flow or not]

If (REAR > n) or (REAR == FRONT)

Print “queue is overflow”

Else

// in next step

[enter the item]

QUEUE[REAR]=value

REAR=REAR+1

End

**Delete an element in queue**

Delete front()

[check for queue is under flow or not]

If front>N or front==Null

Print “queue is empty”

Else

// in next step

If front > rear

Front==null

Rear=-1

**Enter item in Queue:**

Check = Queue is full

If FRONT = 0 and REAR = N-1,

Queue=Overflow and Return

Else

Set REAT = REAT + 1

Set QUEUE[REAR] = ITEM[This inserts new element]

Return

**Delete item in queue**

Check Queue empty

If FRONT=-1

Means (Empty and Return)

Set ITEM = Queue[FRONT] //enter new value in FRONT)

If FRONT = REAR, then [Queue has only one element to start]

Set FRONT = -1 and REAR = -1

Else FRONT = N-1, then

Set FRONT = 0