DS Assignment

Q.1.

Code

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

// (a) Print triangle pattern with 2(n-m+1) lines
void triangle(uint m, const uint &n){
  static bool once = false;
  static const uint _m = m;

  if (m < 0 || n <= 0)
      return;
  if (m > n){
      once = true;
      m--;
  }

  for (int i = 0; i < m; i++)
      cout << '*';
  cout << endl;

  if (once){
      if (m == _m)
          return;
      triangle(m - 1, n);
  }

  else if (m <= n)
      triangle(m + 1, n);
}
```

```
/*(b) TeddyBear game
      Return true if win is possible else false
*/
bool TeddyGame(const int& n)
{
  if (n == 42)
      return 1;

  else if(n > 42)
      if (!(n % 5))
          return TeddyGame(n - 42);

      else if (!(n % 3))
          return TeddyGame(n - n % 10 * (n % 100 / 10));

      else if(!(n % 4)){
          int a = n % 10 * (n % 100 / 10);

          /* Check whether a is 0 or not
              if a == 0 then we can't return (n-a)
              Because it will create a loop with the same value
              that results in a segmentation fault at the end !!!
          */
          if(!a)
              return TeddyGame(n / 2);
          else
              return TeddyGame(n - a);
      }

      else if (!(n % 2))
          return TeddyGame(n / 2);

  return 0;
}
```

```
// (c) character pattern
void pattern(const char &x){
  if (x == 'A')
      cout << x;
  else if (x > 'A'){
      pattern(x - 1);
      cout << x;
      pattern(x - 1);
  }
  else
      return;
}

// (d) Print permutation of string
void permute(string s, int i, const int &n){
  if (i == n - 1)
      cout << "\n\t" << s;
  else
      for (int j = i; j < n; j++){
          swap(s[i], s[j]);
          permute(s, i + 1, n);
          swap(s[i], s[j]);
      }
}

int main(){
  cout << "\n\t\t Question 1 \n";
  cout << "\n(a) Triangle pattern with m = 3 & n = 5 : \n\n";
  triangle(3, 5);
  cout << boolalpha;
  cout <<"\n(b) TeddyBear Game can be won with 250 bears : "
       << TeddyGame(250);
  cout << "\n(b) TeddyBear Game can be won with 252 bears : "
      << TeddyGame(252);
```

```
  cout << "\n\n(c) Character Pattern with D : ";
  pattern('D');

  cout << "\n\n(d) String Permutation of ABC : \n";
  permute("ABC", 0, 3);

  cout << "\n\n";
}
```

Output

```
(base) avinash@avinash-home:~/Desktop/B.SC_CS/Coding/C++$
 Q.1.cpp -o Q.1 && "/home/avinash/Desktop/B.SC_CS/Coding/C

                    Question 1

(a) Triangle pattern with m = 3 & n = 5 :

***
****
*****
*****
****
***

(b) TeddyBear Game can be won with 250 bears : true
(b) TeddyBear Game can be won with 252 bears : false

(c) Character Pattern with D : ABACABADABACABA

(d) String Permutation of ABC :

        ABC
        ACB
        BAC
        BCA
        CBA
        CAB
```

Q.2.

Code

```
#include <iostream>
#include "Sem_3/stack.cpp"
using namespace std;

template<typename q,uint max = 10>
class Queue
{
  private:
      stack_l<q> s1,s2;

  public:
      // Functions
      bool empty(){ return !this->size(); }

      bool full(){ return this->size() == max; }

      uint size(){ return s1.size()+s2.size(); }

      uint capacity(){ return max; }

      void enqueue(const q& val){
          if(this->full())
              throw overflow_error("Queue is full !!!");
          s1.push(val);
      }

     q dequeue(){
          if(this->empty())
              throw underflow_error("Queue is empty !!!");
```

```
        if(s2.empty())
            while(!s1.empty())
                s2.push(s1.pop());
        return s2.pop();
    }

    void detail()
    {
        cout << boolalpha;
        cout << "\nQueue is empty\t : " << this->empty();
        cout << "\nQueue is full\t  : " << this->full();
        cout << "\nSize of Queue\t  : " << this->size();
        cout << "\nCapacity of Queue : "<< this->capacity();
        cout << endl;
    }
};

int main()
{
  Queue<int,5> q1;

  cout << "\nAfter creation, Queue Details : \n";
  q1.detail();

  for(int i = 1; i <= 5; i++)
      q1.enqueue(5*i);

  cout << "\nAfter enqueuing, Queue Details : \n";
  q1.detail();

  cout << "\nYour Queue : ";
  while(!q1.empty())
      cout << q1.dequeue() << ' ';
```

```
  cout << "\n\nAfter dequeuing, Queue Details : ";
  q1.detail();
}
```

Output

```
PROBLEMS    OUTPUT    TERMINAL    DEBUG CONSOLE

(base) avinash@avinash-home:~/Desktop/B.SC_CS/Coding/C++$ cd "/h
sh/Desktop/B.SC_CS/Coding/C++/"Q.2

After creation, Queue Details :

Queue is empty    : true
Queue is full     : false
Size of Queue     : 0
Capacity of Queue : 5


After enqueuing, Queue Details :

Queue is empty    : false
Queue is full     : true
Size of Queue     : 5
Capacity of Queue : 5


Your Queue : 5 10 15 20 25

After dequeuing, Queue Details :
Queue is empty    : true
Queue is full     : false
Size of Queue     : 0
Capacity of Queue : 5
```

Q.3.

Code

```
#include <iostream>
#include "Sem_3/SinglyList.cpp"

using namespace std;

int main(){
  SinglyList<int> s{1,2,5,7,9}, c;
  int sum = 0;

  cout << "\nGiven list of numbers (s) : " << s << endl;
  for(int i:s){
      sum += i;
      c.push(sum);
  }
  cout << "\nCumulative list of s  (c) : " << c << "\n\n";
}
```

Output

```
(base) avinash@avinash-home:~/Desktop/B.SC_CS/Coding/C++$ c
sh/Desktop/B.SC_CS/Coding/C++/"Q.3

Given list of numbers (s) : {1, 2, 5, 7, 9}

Cumulative list of s  (c) : {1, 3, 8, 15, 24}

(base) avinash@avinash-home:~/Desktop/B.SC_CS/Coding/C++$
```

Q.4.

Code

```
#include <iostream>
#include "Sem_3/SinglyList.cpp"
using namespace std;

void set(NODE<int>* a, NODE<int>* b){
  while(a && b){
      a->next = b->next;
      a = a->next;
      if(a)
          b->next = a->next;
      b = b->next;
  }
}

int main(){
  SinglyList<int> s1{1,2,3,4,5};
  NODE<int>* o = s1.begin().base();
  NODE<int>* e = s1.begin().base()->next;

  cout << "\nGiven List of numbers \t\t : " << s1 << endl;

  s1.reset(); // To stop double deletion of a node
  set(o,e);
  SinglyList<int> s2(o);
  SinglyList<int> s3(e);

  cout << "\nList of odd nodes of given list  : " << s2 << endl;
  cout << "\nList of even nodes of given list : " << s3 << endl;
}
```

Output

```
PROBLEMS    OUTPUT    TERMINAL    DEBUG CONSOLE

(base) avinash@avinash-home:~/Desktop/B.SC_CS/Coding/C++$ cd "/hom
sh/Desktop/B.SC_CS/Coding/C++/"Q.4

Given List of numbers            : {1, 2, 3, 4, 5}

List of odd nodes of given list  : {1, 3, 5}

List of even nodes of given list : {2, 4}

(base) avinash@avinash-home:~/Desktop/B.SC_CS/Coding/C++$
```

Q.5.

Code

```
#include <iostream>
#include <fstream>
#include "Sem_3/stack.cpp"
using namespace std;

short prior(const char &c)
{
  switch (c)
  {
  case '+':
  case '-':
      return 1;

  case '*':
  case '/':
  case '%':
      return 2;
  }
  return -1;
}
float calc(const char &c, const float &a, const float &b){
  switch (c)
  {
  case '+':
      return a + b;

  case '-':
      return a - b;

  case '*':
      return a * b;
```

```
  case '/':
      return a / b;

  case '%':
      return int(a) % int(b);
  }
  return 0;
}
bool evaluate(const string &s){
  stack_l<float> val;
  stack_l<char> sign;
  string str = "";

  for (int i = 0, pri = 0; i < s.length(); i++)
  {
      if (s[i] == ' ' || s[i] == ',' || s[i] == '\n')
          continue;

      if (isdigit(s[i]))
      {
          while (isdigit(s[i]) || s[i] == '.')
              str += s[i++];
          val.push(stof(str));
          str = "";
      }

      else if (s[i] == '(')
          sign.push(s[i]);

      else if (s[i] == ')')
      {
          while (!sign.empty() && sign.top() != '(')
          {
              if (val.size() > 1)
                  val.push(calc(sign.pop(), val.pop(), val.pop()));
```

```
            else{
                cerr << "\n Insufficient Variables \n";
                return;
            }
        }
        if (!sign.empty() && sign.top() == '(')
            sign.pop();
        else{
            cerr << "\n Error : Unclosed Parentheses\n";
            return;
        }
    }

    else
    {
        pri = prior(s[i]);

        if (pri == -1)
        {
            cerr << "\n Unknown Operator Found !!!\n";
            return;
        }
        while (!sign.empty() && prior(sign.top()) > pri)
        {
            if (val.size() > 1)
                val.push(calc(sign.pop(), val.pop(), val.pop()));
            else{
                 cerr << "\n Error : Unclosed Parentheses\n";
                 return;
            }
        }
        sign.push(s[i]);
    }
}
```

```
  while (!sign.empty() && !val.empty())
      val.push(calc(sign.pop(), val.pop(), val.pop()));

  cout << "\n Infix Expression : " << s << " = " << val.pop()
       << endl;
}

void f_read(string s){
  fstream fin;
  string data;

  fin.open(s.c_str(), ios::in | ios::binary);

  if (!fin)
      cerr << "Error in opening file !!!";
  else
  {
      while (!fin.eof())
      {
          getline(fin, data);
          evaluate(data);
      }
      fin.close();
  }
}

int main(int argc, char const *argv[])
{
  if (argc == 1 || argc > 2)
      cerr << "Enter a proper file path after program name !!!";
  else
      f_read(argv[1]);
}
```

Output

Text × Q.5.cpp Q.6.cpp Q.7.cpp

C++ > Text

```
1.02 * ( 3 + 1.5 - 0.5 )
2 - 3 * 2
2 & 2
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
(base) avinash@avinash-home:~/Desktop/B.SC_CS/Coding/C+
sh/Desktop/B.SC_CS/Coding/C++/"Q.5 Text

 Infix Expression : 1.02 * ( 3 + 1.5 - 0.5 ) = 4.08

 Infix Expression : 2 - 3 * 2 = -4

 Unknown Operator Found !!!
```

Q.6.

Code

```
#include <iostream>
#include <queue>
#include <stack>
using namespace std;

string s = "";

bool check(queue<int> i, queue<int> o){
  stack <int> tempStack;

  while (!i.empty()){
      int ele = i.front();
      i.pop();
      if (ele == o.front()){
          o.pop();
          s += "\n Enqueue(o_queue, Dequeue(i_queue))";
          while (!tempStack.empty())
          {
              if (tempStack.top() == o.front()){
                  tempStack.pop();
                  o.pop();
                  s += "\n Enqueue(o_queue, Pop(stack))";
              }
              else
                  break;
          }
      }
      else{
          tempStack.push(ele);
          s += "\n Push(stack, Dequeue(i_queue))";
      }
```

```
    }

    return (i.empty() && tempStack.empty());
}

int main()
{
    uint n;
    int v;
    queue<int> input, output;

    cout << "\n Enter the number count : ";
    cin >> n;

    cout << "\n Enter the permutation of numbers...."
         << "\n\n Input Queue : ";

    for(int i = 0; i < n; i++){
        cin >> v;
        input.push(v);
    }
    cout << "\n Output Queue : ";
    for(int i = 0; i < n; i++){
        cin >> v;
        output.push(v);
    }

    if(check(input,output))
        cout << "\n Steps of Stack Permutation : \n" << s << endl;
    else
        cout << "\n Stack Permutation is not possible !!!\n";

}
```

Output

```
PROBLEMS    OUTPUT    TERMINAL    DEBUG CONSOLE

(base) avinash@avinash-home:~/Desktop/B.SC_CS/Coding/C++$ cd "/home/av
sh/Desktop/B.SC_CS/Coding/C++/"Q.6

 Enter the number count : 6

 Enter the permutation of numbers....

 Input Queue : 1 2 3 4 5 6

 Output Queue : 1 3 2 4 5 6

 Steps of Stack Permutation :

 Enqueue(o_queue, Dequeue(i_queue))
 Push(stack, Dequeue(i_queue))
 Enqueue(o_queue, Dequeue(i_queue))
 Enqueue(o_queue, Pop(stack))
 Enqueue(o_queue, Dequeue(i_queue))
 Enqueue(o_queue, Dequeue(i_queue))
 Enqueue(o_queue, Dequeue(i_queue))
```

Q.7.

Code

```
#include <iostream>
#include <iomanip>
#include "Sem_3/SinglyList.cpp"
using namespace std;
```

```
#define no_data(s) if(s.empty()){\
                  cout << "\n Error : No Data Avaliable !!!";\
                  return;}

#define error(b) if(!b){\
                  cout << "\n Error : No match found !!!";\
                  return;}

bool once = 0;
const char getGrade[] = "ABCDF";

class student
{
  private:
    string f_name, l_name, c_code;
    char grade;

  public:

    student(){}

    student(const string& s,const string& s1,const string& s2,
           const  char& c)
    {
          f_name = s;
          l_name = s1;
          c_code = s2;
           grade = c;
    }

    bool chk_code(const string& c) const
    {
           return c == this->c_code;
    }
```

```
    bool chk_l_name(const string& l) const{
         return l == this->l_name;
     }

    short grade_no() const{
         for(short i = 0; i < 5; i++)
              if(this->grade == getGrade[i])
                   return i;
    }

    friend istream& operator>>(istream& in, student& s);

    friend ostream& operator<<(ostream& out, const student& s);

    bool operator==(const student& s){
      return this->f_name == s.f_name && this->l_name == s.l_name
            && this->c_code == s.c_code && this->grade == s.grade;
    }

  ~student(){}
};

void print(SinglyList<student>& l)
{
    no_data(l);
    once = 1;
    for(auto i:l)
         cout << i;
}

void search(SinglyList<student>& f){
    no_data(f);
    string str;
    bool found = 0;
```

```
    cout << "\n\n Enter the last name of student : ";
    cin >> str;
    once = 1;
    for(student i:f)
         if(i.chk_l_name(str)){
              cout << i;
              found = 1;
         }
    error(found);
}

void del(SinglyList<student>& l, student& asp){
    no_data(l);
    error(l.erase(asp));
    cout << "\n Data has been deleted successfully !!!";
}

void avg_class(SinglyList<student>& l){
    no_data(l);
    string str;
    ushort sum = 0, n = 0;
    bool found = 0;

    cout << "\n\n Enter the course code of a class : ";
    cin >> str;

    for(student i:l){
         if(i.chk_code(str)){
              n++;
              sum += i.grade_no();
              found = 1;
         }
    }
    error(found);
```

```
    cout << "\n Class average score of course " + str + " : "
         << getGrade[sum/n] << endl;
}

int main(){
  char op;
  student s;
  SinglyList<student> lst;

  do{
      system("clear");
      cout << "\n<-------------------------------------
                              Menu
              ---------------------------------------->\n";

       cout << "\n\t 1. Insert a new record
              \n\t 2. Delete a record"
          << "\n\t 3. Search the database (by last name)
              \n\t 4. Print records in the database"
          << "\n\t 5. Find the class average for a course
              \n\t 9. Quit \n";
      cout << "\n Enter the option : ";
      cin >> op;
      cout << "\n<-------------------------------------
              ---------------------------------------->\n";

      switch(op){
          case '1':
              cout << "\n<------------------------------
                             Insertion of a Record
                       ------------------------------->\n";
              cin >> s;
              lst.push(s);
              break;
```

```
        case '2':
            cout << "\n<-------------------------------
                           Deletion of a Record
                    ------------------------------->\n";
            cin >> s;
            del(lst,s);
            break;

        case '3':
            cout << "\n<------------------------------
                           Searching of a Record
                    ------------------------------->\n";
            search(lst);
            break;

        case '4':
            cout << "\n<------------------------------
                           Printing all Records
                    -------------------------------->\n";
            print(lst);
            break;

        case '5':
            cout << "\n<------------------------
                   Finding average score of a course
                    ------------------------->\n";
            avg_class(lst);
            break;

        default:
            if(op != '9')
                cout << "\n Wrong Input, Please re-enter !!!";
    }
```

```
        cin.get();
        cin.ignore();

    }while(op != '9');
}

// For the input of the student's details
istream &operator>>(istream &in, student &s)
{
    short c;
    cout << "\n****************************
                Enter Student's Details
            ******************************\n";

    cout << "\tNote:\n"
         << "\t\t1. Course code must not be more than 6 numbers or
                    letters.\n"
         << "\t\t2. For Grading : (1 for A, 2 for B, 3 for C,
                    4 for D, 5 for F)\n\n";
    in.ignore();

    cout << "\n First name \t: ";
    getline(in, s.f_name);

    cout << " Last name  \t: ";
    getline(in, s.l_name);

    cout << " Course Code\t: ";
    getline(in, s.c_code);

    do
    {
        cout << " Grade \t\t: ";
        in >> c;
```

```
        if (c < 1 || c > 5)
        {
            cout << "\n Invalid Entry !!! Please Re-enter !!!\n";
            c = -1;
        }
        else
            s.grade = getGrade[c - 1];
    } while (c == -1);

    cout << "\n*****************************************
                ******************************************\n";
    return in;
}

// For the output of the student's details
ostream &operator<<(ostream &out, const student &s)
{
    if(once)
    {
        out << '\n' << left << setw(25) << " First Name "
            << setw(25) << " Last Name " << setw(17)
            << "Course Code" << setw(10) << "Grade\n";
        once = 0;
    }

    out << "\n  " << left << setw(25) << s.f_name << setw(25)
        << s.l_name << setw(17) << s.c_code << setw(10) << s.grade;
    return out;
}
```

Output

1. Insertion in the student database

```
<------------------------------------- Menu -------------------------------------->

         1. Insert a new record
         2. Delete a record
         3. Search the database (by last name)
         4. Print records in the database
         5. Find the class average for a course
         9. Quit

 Enter the option : 1

<------------------------------------------------------------------------------------>

<------------------------------- Insertion of a Record ------------------------------->

***************************** Enter Student's Details ******************************
        Note:
                1. Course code must not be more than 6 numbers or letters.
                2. For Grading : (1 for A, 2 for B, 3 for C, 4 for D, 5 for F)


 First name     : Avinash
 Last name      : Gautam
 Course Code    : 20021
 Grade          : 1

************************************************************************************
```

```
<------------------------------------- Menu ---------------------------------------->

         1. Insert a new record
         2. Delete a record
         3. Search the database (by last name)
         4. Print records in the database
         5. Find the class average for a course
         9. Quit

 Enter the option : 1

<------------------------------------------------------------------------------------->

<------------------------------- Insertion of a Record ------------------------------->

*************************** Enter Student's Details ******************************
        Note:
                1. Course code must not be more than 6 numbers or letters.
                2. For Grading : (1 for A, 2 for B, 3 for C, 4 for D, 5 for F)


 First name     : Avni
 Last name      : Dubey
 Course Code    : 20022
 Grade          : 2

*************************************************************************************
```

```
<------------------------------------- Menu ---------------------------------------->

         1. Insert a new record
         2. Delete a record
         3. Search the database (by last name)
         4. Print records in the database
         5. Find the class average for a course
         9. Quit

 Enter the option : 1

<------------------------------------------------------------------------------------->

<------------------------------- Insertion of a Record ------------------------------->

**************************** Enter Student's Details ******************************
        Note:
                1. Course code must not be more than 6 numbers or letters.
                2. For Grading : (1 for A, 2 for B, 3 for C, 4 for D, 5 for F)


 First name     : Prince
 Last name      : Sharma
 Course Code    : 20021
 Grade          : 3

*************************************************************************************
```

2. Deletion in Student Database

```
<------------------------------------- Menu ---------------------------------------->

         1. Insert a new record
         2. Delete a record
         3. Search the database (by last name)
         4. Print records in the database
         5. Find the class average for a course
         9. Quit

 Enter the option : 2

<------------------------------------------------------------------------------------>

<------------------------------- Deletion of a Record ------------------------------->

***************************** Enter Student's Details ******************************
        Note:
                1. Course code must not be more than 6 numbers or letters.
                2. For Grading : (1 for A, 2 for B, 3 for C, 4 for D, 5 for F)


 First name     : Avinash
 Last name      : Gautam
 Course Code    : 20021
 Grade          : 1

*************************************************************************************

 Data has been deleted successfully !!!
```

3. Searching in Database

```
<------------------------------------ Menu ------------------------------

         1. Insert a new record
         2. Delete a record
         3. Search the database (by last name)
         4. Print records in the database
         5. Find the class average for a course
         9. Quit

 Enter the option : 3

<-----------------------------------------------------------------------------

<---------------------------- Searching of a Record --------------------

 Enter the last name of student : Dubey

 First Name              Last Name              Course Code      Grade

  Avni                    Dubey                   20022            B
```

4. **Printing the database**

Before Deletion :

```
<------------------------------------ Menu ------------------------------------

         1. Insert a new record
         2. Delete a record
         3. Search the database (by last name)
         4. Print records in the database
         5. Find the class average for a course
         9. Quit

 Enter the option : 4

<-----------------------------------------------------------------------------

<----------------------------- Printing all Records --------------------------

 First Name                Last Name                Course Code      Grade

  Avinash                   Gautam                    20021            A
  Avni                      Dubey                     20022            B
  Prince                    Sharma                    20021            C
```

5. **Finding Class Average of a course**

Before Deletion :

```
<------------------------------------ Menu ------------------------------------

         1. Insert a new record
         2. Delete a record
         3. Search the database (by last name)
         4. Print records in the database
         5. Find the class average for a course
         9. Quit

 Enter the option : 5

<-----------------------------------------------------------------------------

<------------------------ Finding average score of a course -----------------


 Enter the course code of a class : 20021

 Class avarage score of course 20021 : B
```