Name: Ritabroto Ganguly
Roll: 001910501090
OOP Assignment 3

1. Write a macro to find the maximum of two numbers.
Call the macro with
i) two integers as the arguments and
ii) two char * as the arguments. Observe the outcomes.

```
#include<iostream>
#include<cstdlib>
#include<string>
#define max(a,b) a==b ? a : (a<b ? b : a)

using namespace std;

int main()
{
 int a,b;
 string s1,s2;
 cout<<"Enter two values: ";
 cin>>a>>b;
 cout<<"Enter two strings : ";
 cin>>s1>>s2;
 cout<< (max(a,b)) <<" is greater or equal"<<endl;
 cout<<(max(s1,s2))<<" is greater or equal"<<endl;
 return 0;
}
```

```
Enter two values: 1 2
Enter two strings : ola amigo
2 is greater or equal
ola is greater or equal
```

2. Write a function to find the product of two numbers. Call it number of times. Make the functions inline. [check the time of execution and size of object code]

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

inline int product_inline(int a, int b){
 return a*b;
}

int product(int a,int b){
 return a*b;
}

int main()
{
 int a,b;
 cout<<"Enter two value multiplication: ";
 cin>>a>>b;
 clock_t start = clock();
 int result = product(a,b);
 clock_t end = clock();
 double time_taken = ((double)(end-start))/CLOCKS_PER_SEC;
 cout<<"Time taken by normal func="<<time_taken<<endl;
 cout<<result<<endl;

 start = clock();
 result = product_inline(a,b);
 end = clock();
 time_taken = ((double)(end-start))/CLOCKS_PER_SEC;
 cout<<"Time taken by inline func="<<time_taken<<endl;
 cout<<result<<endl;
 return 0;
}
```

```
Enter two value multiplication: 2 3
Time taken by normal func=1.6e-05
6
Time taken by inline func=3e-06
6
```

3. Write a function swap (a, b) to interchange the values of two variables. Do not use pointers.

```
#include <iostream>
#include <cstdlib>
#include <ctime>
#include <vector>

using namespace std;

template<class T>
void my_swap(T& a,T& b){
  T t = move(a);
  a = move(b);
  b = move(t);
}

int main()
{
 vector<int> a(10),b(10);
 for(int i=0;i<10;i++){
     a[i] = i;
     b[i] = i+100;
 }
 cout<<"Before my_swap: "<<endl;
 for(int i=0;i<10;i++)
     cout<<a[i]<<" ";
 cout<<endl;
 for(int i=0;i<10;i++)
     cout<<b[i]<<" ";
 cout<<endl;
 my_swap(a,b);
 cout<<"After my_swap: "<<endl;
 for(int i=0;i<10;i++)
     cout<<a[i]<<" ";
 cout<<endl;
 for(int i=0;i<10;i++)
     cout<<b[i]<<" ";
 cout<<endl;
 return 0;
}
```

```
Before my_swap:
0 1 2 3 4 5 6 7 8 9
100 101 102 103 104 105 106 107 108 109
After my_swap:
100 101 102 103 104 105 106 107 108 109
0 1 2 3 4 5 6 7 8 9
```

4. Write a function max (a, b) that will return the reference of larger value. Store the returned information to x where x is a
i) variable of type a or b,
ii) variable referring to type of a or b. In both the cases modify x. Check also the values of a and b.

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

template<class T>
inline T& my_max(T& a,T& b){
  if(a==b)
    return a;
  else
    if(a>b)
      return a;
    else
      return b;
}

int main()
{
 int a = 1, b = 2;
 int x = my_max<int>(a,b);
 cout<<"Same type:"<<endl;
 cout<<"x="<<x<<endl;
 x+= 10;
 //cout<<“x+=10 ="<<x<<endl;
 cout<<"a="<<a<<" b="<<b<<endl;
 int &y = my_max<int>(a,b);
 cout<<"Refrence type:"<<endl;
 cout<<"y="<<y<<endl;
 y+=10;
 //cout<<“y+=10 ="<<y<<endl;
 cout<<"a="<<a<<" b="<<b<<endl;

 return 0;
}
```

```
Same type:
x=2
a=1 b=2
Refrence type:
y=2
a=1 b=12
```

5. Write a function that will have income and tax rate as arguments and will return tax amount. In case tax rate is not provided it will be automatically taken as 10%. Call it with and without tax rate.

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

inline double tax(int income, double rate = 10.0){
  return income*rate/100;
}

int main()
{
 int income;double rate;
 cout<<"input income and rate(in percent): ";
 cin>>income>>rate;
 double tax_returns = tax(income,rate);
 cout<<fixed<<tax_returns<<endl;
 tax_returns = tax(income);
 cout<<fixed<<tax_returns<<endl;
 return 0;
}
```

```
input income and rate(in percent): 100000000 8
8000000.000000
10000000.000000
```

6. Write a function void f(int) that prints “inside f(int)”. Call the function with actual argument of type:
i) int,
ii) char,
iii) float and
iv) double.
Add one more function f(float) that prints “inside f(float)”. Repeat the calls again and observe the outcomes.

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

```
void f(int i){
  cout<<"inside f(int)"<<endl;
}
void f(char i){
  cout<<"inside f(char)"<<endl;
}

int main()
{
 int a = 1;
 char b = 'b';
 float c = 2.0;
 double d = 3.0;
 //f(a);//f(b);
 //f(c);//f(d);
 return 0;
}
```

7. Define functions f(int, int) and f (char, int). Call the functions with arguments of type (int, char), (char,char) and (float, float).

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

void f(int x,int y){
 cout<<"inside f(int,int)"<<endl;
}
void f(char x,int y){
 cout<<"inside f(char,int)"<<endl;
}

int main()
{
 int a = 1;
 char b = 'b';
 float c = 2.0;
 f(a,b);f(b,b);//f(c,c);
 return 0;
}
```

8. Define a structure student with roll and score as attributes and with two member functions to take input and to show the data. Use the member functions to take data for a structure variable and to show. Write global function i) to modify score and ii) to show the data again.

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

struct student{
  int roll;
  float score;
  void input(){
    cout<<"Enter roll: ";
    int roll;cin>>roll;
    cout<<"Enter score: ";
    int score;cin>>score;
    this->roll = roll;
    this->score = score;
  }
  void show(){
    cout<<"Roll: "<<roll<<endl<<"Score: "<<score<<endl;
  }
}s;

void modify(student &x,int score = s.score,int roll = s.roll){
  x.roll = roll;
  x.score = score;
  x.show();
}

int main()
{
 s.input();
 s.show();
 modify(s,10);
 return 0;
}
```

```
Enter roll: 1
Enter score: 100
Roll: 1
Score: 100
Roll: 1
Score: 10
```

9. Design a class TIME which stores hour, minute and second. The class should have the methods to support the following:
• User may give the time value in 24-hour format.
• User may give the time value in AM/PM format
• Display the time in 24-hour format.

• Display the time in AM/PM format.
• User may like to add minute with a time value.

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

class Time{
  int hour;
  int minute;
  int seconds;
  public:
  void set_time_in_24(int hour,int minute,int seconds){
    this->hour = hour;
    this->minute = minute;
    this->seconds = seconds;
  }
  void set_time_in_ampm(int hr,int minute,int seconds,string s){
    if(s=="AM")
      if(hr==12)
        hour = hr - 12;
      else
        hour = hr;
    else
      if(hr==12)
        hour = hr;
      else
        hour = hr + 12;
    this->minute = minute;
    this->seconds = seconds;
  }
  void show_in_24(){
    cout<<"Time is: "<<hour<<":"<<minute<<":"<<seconds<<endl;
  }
  void show_in_ampm(){
    int hr = hour;
    if(hour>12)
```

```
      hr = hour%12;
    else if(hour==0)
      hr = 12;
    string s = hour>=12 ? "PM" : "AM";
    cout<<"Time is: "<<hr<<":"<<minute<<":"<<seconds<<" "<<s<<endl;
  }
  void add_minutes(int minutes){
    int hr = minutes/60;
    minutes = minutes%60;
    hour = (hour+hr)%24;

    if(minute+minutes>=60){
      hour = (hour+1)%24;
      minute = (minute+minutes)%60;
    }else
      minute += minutes;
  }
};

int main()
{
 Time t;
 t.set_time_in_24(0,55,22);
 t.show_in_24();
 t.show_in_ampm();
 t.add_minutes(200);
 t.show_in_24();
 t.show_in_ampm();
 t.set_time_in_ampm(12,32,11,"PM");
 t.show_in_24();
 t.show_in_ampm();
 t.add_minutes(130);
 t.show_in_24();
 t.show_in_ampm();
 return 0;
}
```

```
Time is: 0:55:22
Time is: 12:55:22 AM
Time is: 4:15:22
Time is: 4:15:22 AM
Time is: 12:32:11
Time is: 12:32:11 PM
Time is: 14:42:11
Time is: 2:42:11 PM
```

10. Create a STACK class with operation for initialization, push and pop. Support for checking underflow and overflow condition are also to be provided.

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

class stack{
  int* arr;
  int size;
  int max;
  public:
  stack(int max_size): arr(new int[max_size]),size(0),max(max_size){}
  void show(){
    cout<<"Stack is:"<<endl;;
    for(int i=size-1;i>=0;i--)
       cout<<arr[i]<<endl;
    cout<<endl;
  }
  void push(int v){
   if(size<max)
      arr[size++] = v;
   else
     cout<<"Stack is already full"<<endl;
  }
  void pop(){
   if(size>0)
     size--;
   else
     cout<<"Stack is already empty"<<endl;
  }
  bool is_empty(){
    return (!size);
  }
  int curr_size(){
    return size;
  }
};

int main()
{
 stack s(5);
 cout<<s.is_empty()<<endl;
```

```
 cout<<s.curr_size()<<endl;
 for(int i=0;i<6;i++){
    s.push(i+12);
 }
 s.show();
 s.pop();
 cout<<s.is_empty()<<endl;
 cout<<s.curr_size()<<endl;
 s.show();
 for(int i=0;i<5;i++)
    s.pop();
 s.show();
 s.push(100);
 cout<<s.is_empty()<<endl;
 cout<<s.curr_size()<<endl;
 s.show();
 return 0;
}
```

```
1
0
Stack is already full
Stack is:
16
15
14
13
12

0
4
Stack is:
15
14
13
12

Stack is already empty
Stack is:

0
1
Stack is:
100
```

11. Create an APPLICANT class with application id (auto generated as last id +1), name and score. Support must be there to receive applicant data, show applicant details and to find out number of applicants.

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

class applicants{
  static int num_of_app;
  string name;
  int score;
  int roll;
  public:
  void admit(const string &name,const int &score){
   this->name = move(name); this->score = score;
   num_of_app++;
   roll = num_of_app;
  }
  void show(){cout<<"Roll: "<<roll<<endl<<"Name:
"<<name<<endl<<"Score: "<<score<<endl;}
  static void number_of_applicants(){cout<<"Number of applicants is:
"<<num_of_app<<endl;}
};
int applicants::num_of_app = 0;

int main()
{
 applicants a,b;
 a.admit("Rahul Maiti",100);
 a.show();
 applicants::number_of_applicants();
 b.admit("Rishav paul",90);
 b.show();
 applicants::number_of_applicants();

 return 0;
}
```

```
Roll: 1
Name: Rahul Maiti
Score: 100
Number of applicants is: 1
Roll: 2
Name: Rishav paul
Score: 90
Number of applicants is: 2
```

12. Design a STUDENT class to store roll, name, course, admission date and marks in 5 subjects. Provide methods corresponding to admission (marks are not available then), receiving marks and preparing mark sheet. Support must be there to show the number of students who have taken admission.

```
#include <iostream>
#include <cstdlib>
#include <ctime>
using namespace std;
const int subs = 5;
const string subjects[5] = {"Maths","OOP","DSA","COA","Digital Logic"};

class student{
  static int num;
  string name;
  float scores[subs];
  int roll;
  string course;
  string date;
  void marksheet(){
   cout<<"Date of admission: "<<date<<endl;
   cout<<"Roll: "<<roll<<endl<<"Name: "<<name<<endl<<"Course:
"<<course<<endl;
   for(int i=0;i<subs;i++)
      cout<<subjects[i]<<" - "<<scores[i]<<endl;
  }
  public:
  void admit(const string &name,const string& course,const string& date){
   this->name = move(name);
   num++;
   roll = num;
   this->course = move(course);
   this->date = move(date);
  }
  void give_marks(){
    cout<<name<<"'s "<<"scores:"<<endl;
    for(int i=0;i<subs;i++){
       cout<<"Enter marks for "<<subjects[i]<<": ";
       cin>>scores[i];
    }
    marksheet();
  }
  static void number_of_students(){cout<<"Number of students is:
"<<num<<endl;}
};
```

```
int student::num = 0;

int main()
{
 student a,b;
 a.admit("Rahul Maiti","CSE","22/12/23");
 a.give_marks();
 student::number_of_students();
 b.admit("Rishav paul","IEE","20/10/23");
 b.give_marks();
 student::number_of_students();

 return 0;
}
```

```
Rahul Maiti's scores:
Enter marks for Maths: 100
Enter marks for OOP: 90
Enter marks for DSA: 99
Enter marks for COA: 99
Enter marks for Digital Logic: 98
Date of admission: 22/12/23
Roll: 1
Name: Rahul Maiti
Course: CSE
Maths - 100
OOP - 90
DSA - 99
COA - 99
Digital Logic - 98
Number of students is: 1
Rishav paul's scores:
Enter marks for Maths: 78
Enter marks for OOP: 78
Enter marks for DSA: 78
Enter marks for COA: 88
Enter marks for Digital Logic: 88
Date of admission: 20/10/23
Roll: 2
Name: Rishav paul
Course: IEE
Maths - 78
OOP - 78
DSA - 78
COA - 88
Digital Logic - 88
Number of students is: 2
```

13. Create a class for linked list. Consider a separate class NODE for basic node activities and use it in class for linked list.

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

class node{
 int s;
 node *next;
 node *head;
 node *tail;
 public:
 static int nodes;
 void create(int x);
 void insert(int x,int pos);
 void remove(int pos);
 void print_list(){
  node *h = head;
  for(;h!=NULL;h=h->next){
      cout<<h->s<<"->";
  }
  cout<<"NULL"<<endl;
 }
};

class List{
 node n;
 public:
 List(int v){n.create(v);}
 void push_back(int v){n.insert(v,(node::nodes)+1);}
 void push_front(int v){n.insert(v,1);}
 void insert(int v,int pos){n.insert(v,pos);}
 void pop_back(){n.remove(node::nodes);}
 void pop_front(){n.remove(1);}
 void remove(int pos){n.remove(pos);}
 void display(){n.print_list();}
 int count(){return node::nodes;}
};

int node::nodes = 0;

int main()
{
 List l(11);
 for(int i=0;i<5;i++)
```

```
    l.push_back(i+20);
 l.display();cout<<l.count()<<endl;
 for(int i=0;i<5;i++)
    l.push_front(i+200);
 l.display();cout<<l.count()<<endl;
 for(int i=0;i<5;i++)
    l.pop_front();
 l.display();cout<<l.count()<<endl;
 for(int i=0;i<5;i++)
    l.pop_back();
 l.display();cout<<l.count()<<endl;

 for(int i=0;i<5;i++)
    l.insert(i*2,i+2);
 l.display();cout<<l.count()<<endl;
 for(int i=0;i<5;i++)
    l.insert(i*10,i+4);
 l.display();cout<<l.count()<<endl;
 for(int i=0;i<5;i++)
    l.remove(i);
 l.display();cout<<l.count()<<endl;
 return 0;
}

void node::create(int x){
 node *newNode = new node;
 newNode->s = x;
 newNode->next = NULL;
 head = newNode;
 tail = head;
 node::nodes++;
}
void node::insert(int x,int pos){
 if(pos<1){//illegal position
   printf("No such Position!\n");
   return;
 }

 if(pos>(nodes+1)){
   cout<<"Total number of nodes present is "<<nodes<<" so appending
nodes to the current list at "<<(nodes+1)<<" position"<<endl;
 }

 if(nodes==0){
    node::create(x);
```

```
    return;
  }

  if(pos>=(nodes+1)){//append to the end of the list
    node *newNode = new node;
    newNode->s = x;
    newNode->next = NULL;
    tail->next = newNode;
    tail = newNode;
    node::nodes++;
    return;
  }

  /*------Insert in middle or front------*/
  node *temp = head;
  int traversed = 1;
  while(traversed<(pos-1)){
       temp = temp->next;
       traversed++;
  }
  node *newNode = new node;
  newNode->s = x;
  if(pos>1){
    newNode->next = temp->next;
    temp->next = newNode;
  }
  else{
    newNode->next = head;
    head = newNode;
    if(nodes<1)
      tail = head;
  }
  node::nodes++;
}

void node::remove(int pos){
  if(pos>nodes || pos<1){//illegal position
    printf("No such data!\n");
    return;
  }
  node *temp = head;
  if(pos==1){//deleting first node
    head = head->next;
    delete temp;
    node::nodes--;
```

```
    return;
  }

 /*-----deleting any other node than first node-----*/
  int traversed = 1;
  while(traversed<(pos-1)){
       temp = temp->next;
       traversed++;
  }
  if(temp->next->next==NULL)
     tail = temp;
  node *del = temp->next;
  temp->next = temp->next->next;
  delete del;
  node::nodes--;
 }
```

```
11->20->21->22->23->24->NULL
6
204->203->202->201->200->11->20->21->22->23->24->NULL
11
11->20->21->22->23->24->NULL
6
11->NULL
1
11->0->2->4->6->8->NULL
6
11->0->2->0->10->20->30->40->4->6->8->NULL
11
No such data!
0->0->20->40->4->6->8->NULL
7
```

14. Design the class(es) for the following scenario:
• An item list contains item code, name, rate, and quantity for several items.
• Whenever a new item is added in the list uniqueness of item code is to be checked.
• Time to time rate of the items may change.
• Whenever an item is issued or received existence of the item is checked and quantity is updated.
• In case of issue, availability of quantity is also to be checked.
• User may also like to know price/quantity available for an item.

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

struct item{
  int code;
  string name;
  float rate;
  int qty;
};
ostream& operator<<(ostream& o,const item& i){
  o<<i.code<<" "<<i.name<<" "<<i.rate<<" "<<i.qty;
  return o;
}

class item_list{
  public:
  vector<item> vi;
  void add_to_vi(const item& i){vi.push_back(i);}
  void print_list(){
   for(auto itr = vi.begin();itr!=vi.end();++itr)
      cout<<*itr<<endl;
  }
  item* find_item(int c){
    for(auto itr = vi.begin(); itr!=vi.end();++itr)
       if(itr->code==c)
         return &(*itr);
    return nullptr;
  }
};

class User{
  item_list il;
  public:
  void add_item(int c,const string& n,float r,int q){
     int in = c;
     item *x = il.find_item(c);
     while(x!=nullptr && x->name!=n){//code is already present with a product
of a different name.
       cout<<"Item already present: "<<*x<<endl<<"Enter a valid code(Press
-1 to cancel this product): ";
       cin>>in;
       x = il.find_item(in);
     }
     if(in==-1)//cancel this product
```

```
      return;

    if(x!=nullptr && x->name==n)//over write the previous product of same
name and code.
       il.vi.erase(il.vi.begin()+(x-&(il.vi[0])));

    item i = {in,n,r,q};
    il.add_to_vi(i);
  }
  void add_item(item& i){
    add_item(i.code,i.name,i.rate,i.qty);
  }

  void change_rate(int code,float rate){
    item* x = il.find_item(code);
   if(x)
    x->rate = rate;
  }

  item* check_availability(int code){
   item* x;
   if((x=il.find_item(code))){
     cout<<*x<<endl;
     return x;
   }else{
     cout<<"No such item in inventory"<<endl;
     return nullptr;
   }
  }

  void issue_item(int code,int num){
    item* x = check_availability(code);
    if(x!=nullptr && x->qty>=num){
      x->qty-=num;
    }
  }

  void print_list(){il.print_list();}
};

int main()
{
 User user;
 item i = {1001,"kola",22.50,10};
 user.add_item(i);
```

```
 user.issue_item(1001,3);
 user.change_rate(1001,24);
 user.check_availability(1001);
 item b = {1001,"kola",28.50,12};
 user.add_item(b);
 user.issue_item(1002,10);
 user.issue_item(1001,1);
 user.add_item(1001,"ola",23,20);
 user.print_list();
 return 0;
}
```

```
1001 kola 22.5 10
1001 kola 24 7
No such item in inventory
1001 kola 28.5 12
Item already present: 1001 kola 28.5 11
Enter a valid code(Press -1 to cancel this product): 10
1001 kola 28.5 11
1002 ola 23 20
```

15. Design a BALANCE class with account number, balance and date of last update. Consider a TRANSACTION class with account number, date of transaction, amount and transaction type (W for withdrawal and D for deposit). If it is a withdrawal check whether the amount is available or not. Transaction object will make necessary update in the BALANCE class.

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

enum type{W,D,I};
ostream& operator<<(ostream& o,const type& t){
  if(t==0)
    o<<"Withdrawal";
  else if(t==1)
    o<<"Deposit";
  else
    o<<"Incomplete";
  return o;
}
```

```
class balance{
  string last_upd;
  int acc;
  float bal;
  public:
  balance(int acc,const string& d,int bal):acc(acc),bal(bal),last_upd(d){}
  void set_bal(int x){bal=x;}
  float get_bal(){return bal;}
  int get_acc(){return acc;}
  void show_balance(){cout<<acc<<": "<<last_upd<<"\t"<<"Rupees
"<<bal<<endl;}
};

class transaction{
  int acc;
  float aot;
  type t;
  string dot;
  public:
  transaction(int acc,const string& d,float a,type t):acc(acc),dot(d),aot(a),t(t){}
  string get_dot(){return dot;}
  void deposit(balance* b){b->set_bal(aot);}
  void withdraw(balance *b){if(b->get_bal()>=aot)b->set_bal((b->get_bal()-
aot));else{cout<<"Insufficient balance"<<endl;t=I;}}
  void show_transaction(){cout<<acc<<": "<<dot<<"\tRupees
"<<aot<<"\t"<<t<<endl;}
};

class account{
  vector<transaction> t;
  balance *b;
  static int acc;
  public:
  account(const string& d,int bal=0){b = new balance(acc,d,bal);acc++;}
  void transact(int type,float amt,const string& d){
    transaction t1(b->get_acc(),d,amt,static_cast<enum type>(type));
    if(type==W){
      t1.withdraw(b);
    }else{
      t1.deposit(b);
    }
    t.push_back(t1);
  }
```

```
  void find_transactions(const string& d){
    b->show_balance();
    int c = 0;
    for(auto itr = t.begin();itr!=t.end();++itr)
      if(itr->get_dot()==d){
        itr->show_transaction();
        c++;
      }
    if(c>0)
      cout<<c<<" transaction(s) was/were made on "<<d<<endl;
    else
      cout<<"No such transaction"<<endl;
  }
  void list_transactions(){
    b->show_balance();
    for(auto itr = t.begin();itr!=t.end();++itr)
        itr->show_transaction();
  }
};

int account::acc = 1000;

int main()
{
 account a("22/10/23");
 a.transact(W,10000,"23/10/23");
 //a.transact(D,10000,"23/10/23");
 a.transact(W,9000,"24/10/23");
 a.find_transactions("24/10/23");
 return 0;
}
```

```
Insufficient balance
Insufficient balance
1000: 22/10/23  Rupees 0
1000: 24/10/23  Rupees 9000     Incomplete
1 transaction(s) was/were made on 24/10/23
```