Name:- \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Answer the following Questions:

**Q1: [4 points]** For the following T(n) find values of n0 and c such that cn^3 is larger than T(n) for all n larger than n0.

T(n) = n^3 – 5n^2 + 20n – 10

|  |
| --- |
| Answer the question using this space.  C =35 , n0 = 1 |

**Q2: [6 points]** For the C++ function shown below, find the T(n) and find values of n0 and c.

**int selectkth(int a[], int k, int n) {**

**int i, j, mini, tmp;**

**for (i = 0; i < k; i++) {**

**mini = i;**

**for (j = i+1; j < n; j++)**

**if (a[j]<a[mini])**

**mini = j;**

**tmp = a[i];**

**a[i] = a[mini];**

**a[mini] = tmp;**

**}**

**return a[k-1];**

**}**

|  |
| --- |
| Answer the question using this space. |

**Q3: [9 points]** Develop an OOP program to maintain a list of homework assignments. When an assignment is assigned, add it to the list, and when it is completed, remove it. You should keep track of the due date. Your program should provide the following services:

• Add a new assignment.

• Remove an assignment.

• Provide a list of the assignments in the order they were assigned.

• Find the assignment(s) with the earliest due date.

create a class called assignments to complete the task and use STL to implement the linked list. compile and run your project and test all the methods.

|  |
| --- |
| Provide your code and the results of running the project using this space  #include <iostream>  #include <string>  #include <list>  #include <iterator>  using namespace std;  struct dueDate {      int month, day, year;      // override the == operator      friend bool operator==(dueDate a, dueDate b) {          if (a.month=b.month && a.day==b.day && a.year==b.year) {              return true;          }          else {              return false;          }      }  };  class Assignment  {  private:      string assignment\_name;      dueDate due\_date;  public:      Assignment(string assignment\_name, dueDate due\_date);      dueDate get\_due\_date();      string get\_name();      ~Assignment();  };  Assignment::Assignment(string assignment\_name, dueDate due\_date)  {      this->assignment\_name = assignment\_name;      this->due\_date = due\_date;  }  dueDate Assignment::get\_due\_date()  {      return this->due\_date;  }  string Assignment::get\_name()  {      return this->assignment\_name;  }  Assignment::~Assignment()  {  }  list <Assignment> assignments;  void add\_assignment(string assignment\_name, dueDate due\_date)  {      Assignment assignment(assignment\_name, due\_date);      assignments.push\_back(assignment);  }  void del\_assignment(string assignment\_name, list <Assignment> &a)  {      list <Assignment> :: iterator it;      for(it = a.begin(); it != a.end(); it++)      {          if (it->get\_name() == assignment\_name)          {              it = a.erase(it);          }      }  }  void list\_assignments(list <Assignment> a)  {      list <Assignment> :: iterator it;      for (it = a.begin(); it != a.end(); it++)      {          cout << it->get\_name() << "\t";      }      cout << endl;  }  dueDate next\_due(list<Assignment> loA)  {      dueDate temp;      int mon = loA.front().get\_due\_date().month;      for (auto f = loA.begin(); f != loA.end(); f++) {          if (mon > f->get\_due\_date().month)              mon = f->get\_due\_date().month;      }      list<Assignment> DDt;      for (auto f = loA.begin(); f != loA.end(); f++) {          if (mon == f->get\_due\_date().month)              DDt.push\_back(\*f);      }      auto f = DDt.begin();      temp = f->get\_due\_date();      f++;      for (;f!=DDt.end();f++) {          if (f->get\_due\_date().day < temp.day)              temp = f->get\_due\_date();      }      return temp;  }  int main()  {      dueDate date;      date.day = 15;      date.month = 9;      date.year = 2021;      add\_assignment("HW 1", date);      date.day = 29;      add\_assignment("HW 2", date);      date.day = 1;      add\_assignment("HW 3", date);      date.day = 8; date.month = 8;      add\_assignment("HW 4", date);      list\_assignments(assignments);      dueDate due\_date = next\_due(assignments);      int day = due\_date.day, month = due\_date.month, year = due\_date.year;      cout << day << ", " << month << ", " << year << endl;      del\_assignment("HW 3", assignments);      list\_assignments(assignments);      return 0;  }  Output: HW 1 HW 2 HW 3 HW 4  8, 8, 2021  HW 1 HW 2 HW 4 |

**Q4: [6 points]** For the double-linked list in the figure below, explain the effect of each statement in the following fragments. **MAKE SURE TO STATE THE FINALE RESULT OF EACH FRAGMENT.** Each fragment should be treated independently.

a. DNode\* node\_ptr = tail->prev;

node\_ptr->prev->next = tail;

tail->prev = node\_ptr->prev;

|  |
| --- |
| Answer Q4: a using this space.  New pointer is created and is pointing at the last pointer in the list with information;  Sets the new nodes next pointer to point at the tail pointer to keep track of the end of list;  The tail prev pointer is assigned to point at the location of the new node;  Node\_ptr has been added to the back of a double linked list |

b. DNode\* node\_ptr = head;

head = new DNode("Tamika");

head->next = node\_ptr;

node\_ptr->prev = head;

|  |
| --- |
| Answer Q4: b using this space.  node\_ptr is created pointing at the location of head;  sets the value at the location of head to Tamika;  head points to node\_ptr as its next pointer  sets the node\_ptr previous pointer to the head pointer;  Tamika was added to the beginning of the linked list |

c. DNode\* node\_ptr= new DNode(“Adam”);

tail->prev->next=node\_ptr;

node\_ptr->prev=tail->prev;

tail=node\_ptr;

|  |
| --- |
| Answer Q4: c using this space.  Node\_ptr created pointing at the value Adam;  Makes the last pointer in the list point and the newly created node\_ptr;  Node\_ptr previous pointer is pointed at the old last node;  Tail is now pointing to the same location as node\_ptr, location with value Adam  Adam was added to the end of the linked list. |

