ASSIGNMENT-LAB 03

Course Code: CSE - 2322 **Course Title:** Data Structures (& Lab)
Course Teacher: Mohammed Shamsul Alam

Name: Ezaz Ahmed **ID:** C223009 **Section:** 3AM

Problem 01: **Write a program to calculate the Factorial of a number using recursive and non-recursive method.**

Answer:

```
#include <bits/stdc++.h>
using namespace std;

int norec(int n)
{
   int f = 1;
   for (int i = 1; i <= n; i++)
   {
      f *= i;
   }
   return f;
}
int recr(int n)
{
   if (n == 1)
   {
      return 1;
   }
   return n * recr(n - 1);
}
int main()
{
   int k;
   cin >> k;

   cout << "Factorial using recursive method: " << recr(k) << endl;
   cout << "Factorial using non-recursive method: " << norec(k) << endl;
   return 0;
}
```

Problem 02: Write a program to find the nth term Fn of the Fibonacci sequence using recursive and non-recursive method.

Answer:

```
#include <bits/stdc++.h>
using namespace std;

int nrec(int n)
{
   int a = 0, b = 1;
   if (n == 0)
   {
      return a;
   }
   for (int i = 2; i <= n; i++)
   {
      int temp = a + b;
      a = b;
      b = temp;
   }
   return b;
}
int rec(int n)
{
   if (n <= 1)
   {
      return n;
   }
   return rec(n - 1) + rec(n - 2);
}
int main()
{
   int n;
   cout << "Enter the value of n: ";
   cin >> n;

   int r1 = rec(n);
   int r2 = nrec(n);
```

```
    cout << n <<"th term of fibonacci using recursive method: " << r1 << endl;
    cout << n <<"th term of fibonacci using non recursive method: " << r2 << endl;

    return 0;
}
```

Problem 03: Write a program to move n disks for Tower of Hanoi problem.

Answer:

```
#include <bits/stdc++.h>
using namespace std;
long long TMo(long long  N)
{
    long long  d = 1;
    for (long long  i = 0; i < N; i++)
    {
        d *= 2;
    }
    return d-1;
}
void TOH(long long  N, string S, string Aux, string Des)
{
    if (N == 1)
    {
        cout << "Move disk 1 from " << S << " to " << Des << endl;
        return;
    }

    TOH(N - 1, S, Des, Aux);
    cout << "Move disk " << N << " from " << S << " to " << Des << endl;

    TOH(N - 1, Aux, S, Des);
}

int main()
{
    long long  N;
```

```
    cout << "Enter the number of disks: ";
    cin >> N;
    cout<<"Total Moves: "<<TMo(N)<<endl;
    TOH(N, "Source", "Auxiliary", "Destination");
    return 0;
}
```

Problem 04: Write a program to find the value from Ackerman function.

Answer:

```
#include <bits/stdc++.h>
using namespace std;
int ak(int c, int d)
{
    if (c == 0)
    {
        return d + 1;
    }
    else if((c > 0) && (d == 0))
    {
        return ak(c - 1, 1);
    }
    else if((c > 0) && (d > 0))
    {
        return ak(c - 1, ak(c, d - 1));
    }
}

int main()
{
    int a, b;
    cout << "Enter the values of a and b: ";
    cin >> a >> b;
    int r = ak(a, b);
    cout << "Ackerman Function of "<<a<<" and "<<b<<" is "  << r << endl;
    return 0;
}
```

Problem 05: Write a program to show the insert and delete operations of a circular queue.

Answer:

```
#include <bits/stdc++.h>
using namespace std;

#define N 5
int queue[N + 1], front = 0, rear = 0;

int menu(void)
{
   int choice;
   do
   {
      cout << "\n1-Enqueue\n2-Dequeue\n0-Exit\n";
      cout << "Enter your choice: ";
      cin >> choice;

      if (choice < 0 || choice > 2)
      {
         cout << "\nWrong...Choice again...\n";
      }
   }
   while (choice < 0 || choice > 2);
   return choice;
}

void Qins(int value)
{
   if ((front == 1 && rear == N) || (front == rear + 1))
   {
      cout << "Overflow. Cannot enqueue.\n";
      return;
   }
   if (front == 0)
   {
      front = rear = 1;
```

```
    }
    else if (rear == N)
    {
        rear = 1;
    }
    else
    {
        rear++;
    }
    queue[rear] = value;
}

void Qdlt()
{
    if (front == 0)
    {
        cout << "Queue is empty. Cannot dequeue.\n";
        return;
    }
    cout << "Dequeued element: " << queue[front] << endl;
    if (front == rear)
    {
        front = rear = 0;
    }
    else if (front == N)
    {
        front = 1;
    }
    else
    {
        front++;
    }
}

void display()
{
    if (front == 0)
```

```
    {
        cout << "Queue is empty.\n";
        return;
    }
    cout << "Queue contents: ";
    if (front <= rear)
    {
        for (int i = front; i <= rear; i++)
        {
            cout << queue[i] << " ";
        }
    }
    else
    {
        for (int i = front; i <= N; i++)
        {
            cout << queue[i] << " ";
        }
        for (int i = 1; i <= rear; i++)
        {
            cout << queue[i] << " ";
        }
    }
    cout << "\n";
}

int main()
{
    int choice;
    do
    {
        choice = menu();
        switch (choice)
        {
        case 1:
        {
            int value;
```

```
            cout << "Enter the value to enqueue: ";
            cin >> value;
            Qins(value);
            display();
        }
        break;
        case 2:
            Qdlt();
            display();
            break;
        case 0:
            cout << "End of operation\n";
            break;
        }
    }
    while (choice != 0);
    return 0;
}
```

Problem 06: Write a program to show the insert and delete operations of a priority queue using linked-list.

Answer:

```
#include <bits/stdc++.h>
using namespace std;

struct link
{
    int data;
    int pri;
    link* n;
};

link* f = nullptr;

void ins()
{
    int d, p;
    cout << "Enter data: ";
```

```
    cin >> d;
    cout << "Enter priority: ";
    cin >> p;

    link* x = new link;
    x->data = d;
    x->pri = p;
    x->n = nullptr;

    if (f == nullptr || p < f->pri)
    {
        x->n = f;
        f = x;
    }
    else
    {
        link* prev = nullptr;
        link* curr = f;
        while (curr != nullptr && p >= curr->pri)
        {
            prev = curr;
            curr = curr->n;
        }

        x->n = curr;
        prev->n = x;
    }
}

void dlt()
{
    if (f == nullptr)
    {
        cout << "Linked List is empty. Cannot delete." << endl;
        return;
    }
    link* t = f;
```

```
    f = f->n;
    delete t;
}

void dis()
{
    if (f == nullptr)
    {
        cout << "Linked List is empty." << endl;
        return;
    }
    link* t = f;
    cout << "Linked List elements (data, priority): ";
    while (t != nullptr)
    {
        cout << "(" << t->data << ", " << t->pri << ") ";
        t = t->n;
    }
    cout << endl;
}

int main()
{
    int c;
    do
    {
        cout << "\n1-Insert\n2-Delete\n0-Exit\n";
        cout << "Enter your choice: ";
        cin >> c;

        switch (c)
        {
        case 1:
            ins();
            dis();
            break;
        case 2:
```

```
            dlt();
            dis();
            break;
        case 0:
            cout << "End of operations.\n";
            break;
        default:
            cout << "Invalid choice. Try again.\n";
            break;
        }
    }
    while (c != 0);

    return 0;
}
```

Problem 07: Write a program to show the insert and delete operations of a priority queue using array.

Answer:

```
#include <bits/stdc++.h>
using namespace std;

#define N 5

int pq[N];
int size = 0;

int menu()
{
    int choice;
    do
    {
        cout << "\n1-Enqueue\n2-Dequeue\n0-Exit\n";
        cout << "Enter your choice: ";
        cin >> choice;

        if (choice < 0 || choice > 2)
        {
```

```
            cout << "\nWrong choice. Try again.\n";
        }
    }
    while (choice < 0 || choice > 2);
    return choice;
}

void enqueue(int value)
{
    if (size == N)
    {
        cout << "Priority queue is full. Cannot enqueue.\n";
        return;
    }

    int i = size;
    while (i > 0 && value > pq[(i - 1) / 2])
    {
        pq[i] = pq[(i - 1) / 2];
        i = (i - 1) / 2;
    }

    pq[i] = value;
    size++;
}

void dequeue()
{
    if (size == 0)
    {
        cout << "Priority queue is empty. Cannot dequeue.\n";
        return;
    }

    int hp = pq[0];
    size--;
```

```
    int i = 0;
    int j = 2 * i + 1;

    while (j < size)
    {
        if (j + 1 < size && pq[j + 1] > pq[j])
        {
            j++;
        }

        if (pq[j] > pq[i])
        {
            pq[i] = pq[j];
            i = j;
            j = 2 * i + 1;
        }
        else
        {
            break;
        }
    }

    pq[i] = pq[size];
    cout << "Dequeued element with highest priority: " << hp << endl;
}

void display()
{
    if (size == 0)
    {
        cout << "Priority queue is empty.\n";
        return;
    }
    cout << "Priority queue contents: ";
    for (int i = 0; i < size; i++)
    {
        cout << pq[i] << " ";
```

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

int main()
{
    int choice;
    do
    {
        choice = menu();
        switch (choice)
        {
        case 1:
        {
            int value;
            cout << "Enter the value to enqueue: ";
            cin >> value;
            enqueue(value);
            display();
            break;
        }
        case 2:
            dequeue();
            display();
            break;
        case 0:
            cout << "End of operations.\n";
            break;
        }
    }
    while (choice != 0);

    return 0;
}
```

Problem 08: Write a program to create a Linked List of n elements and then display the list.

Answer:

```
#include <bits/stdc++.h>
using namespace std;

struct linked_list {
   int num;
   linked_list* next;
};
typedef linked_list node;

int main() {
   int n, i, item, ele;
   node* start, * ptr;

   start = (node*)malloc(sizeof(node));
   ptr = start;

   cout << "How many elements: ";
   cin >> n;

   for (i = 1; i <= n; i++) {
      cout << "Input a number: ";
      cin >> ptr->num;
      if (i != n) {
         ptr->next = (node*)malloc(sizeof(node));
         ptr = ptr->next;
      }
   }
   ptr->next = nullptr;

   cout << "\nElements in the Linked list are: \n";
   ptr = start;
   while (ptr != nullptr) {
      cout << ptr->num << endl;
      ptr = ptr->next;
   }

   return 0;
```

}

Problem 09: Write a program to create a Linked List of n elements and then search an element from the list.

Answer:

```
#include<bits/stdc++.h>
using namespace std;

struct linked_list
{
   int num;
   linked_list* next;
};
typedef linked_list node;

int main()
{
   int n, i, item, ele;
   node* start, * ptr;

   start = (node*)malloc(sizeof(node));
   ptr = start;

   cout << "How many elements: ";
   cin >> n;

   for (i = 1; i <= n; i++)
   {
      cout << "Input a number: ";
      cin >> ptr->num;
      if (i != n)
      {
         ptr->next = (node*)malloc(sizeof(node));
         ptr = ptr->next;
      }
   }
   ptr->next = nullptr;
   cout << "\nElements in the Linked list are: \n";
```

```
    ptr = start;
    while (ptr != nullptr)
    {
        cout << ptr->num << endl;
        ptr = ptr->next;
    }
    cout << "Enter the element to search: ";
    cin >> ele;

    ptr = start;
    while (ptr != nullptr)
    {
        if (ptr->num == ele)
        {
            cout << ele <<" is found."<< endl;
            break;
        }
        ptr = ptr->next;
    }
    if (ptr == nullptr)
    {
        cout <<ele << " is not found." << endl;
    }

    return 0;
}
```

Problem 10: Write a program to create a Linked List of n elements and then insert an element to the list.

Answer:

```
#include <bits/stdc++.h>
using namespace std;

struct link
{
    int data;
```

```
    link* next;
};

int main()
{
    int n, i, item, ele;

    link* h = nullptr;
    link* t = nullptr;
    link* avail = nullptr;

    cout << "How many elements: ";
    cin >> n;

    for (i = 1; i <= n; i++)
    {
        cout << "Input a number: ";
        cin >> item;

        link* newNode;

        if (avail != nullptr)
        {
            newNode = avail;
            avail = avail->next;
        }
        else
        {
            newNode = new link;
        }

        newNode->data = item;
        newNode->next = nullptr;

        link* prev = nullptr;
        link* current = h;
```

```
   while (current != nullptr && current->data < item)
   {
      prev = current;
      current = current->next;
   }

   if (prev == nullptr)
   {
      newNode->next = h;
      h = newNode;
   }
   else
   {
      prev->next = newNode;
      newNode->next = current;
   }
}

cout << "\nElements in the sorted list: \n";
link* c = h;
while (c != nullptr)
{
   cout << c->data << " ";
   c = c->next;
}
cout << endl;

cout << "Enter the element to insert: ";
cin >> ele;

link* newNode;

if (avail != nullptr)
{
   newNode = avail;
   avail = avail->next;
}
```

```
else
{
   newNode = new link;
}

newNode->data = ele;
newNode->next = nullptr;

link* prev = nullptr;
link* current = h;

while (current != nullptr && current->data < ele)
{
   prev = current;
   current = current->next;
}

if (prev == nullptr)
{
   newNode->next = h;
   h = newNode;
}
else
{
   prev->next = newNode;
   newNode->next = current;
}

cout << "\nElements in the sorted list after insertion: \n";
c = h;
while (c != nullptr)
{
   cout << c->data << " ";
   c = c->next;
}

while (h != nullptr)
```

```
    {
        link* temp = h;
        h = h->next;
        temp->next = avail;
        avail = temp;
    }

    while (avail != nullptr)
    {
        link* temp = avail;
        avail = avail->next;
        delete temp;
    }

    return 0;
}
```

Problem 11: Write a program to create a Linked List of n elements and then delete an element from the list.

Answer:

```
#include <bits/stdc++.h>
using namespace std;

struct link
{
    int data;
    link* next;
};

int main()
{
    int n, i, item, ele;

    link* h = nullptr;
    link* t = nullptr;

    cout << "How many elements: ";
    cin >> n;
```

```
for (i = 1; i <= n; i++)
{
   cout << "Input a number: ";
   cin >> item;

   link* n = new link;
   n->data = item;
   n->next = nullptr;

   if (t != nullptr)
   {
      t->next = n;
   }

   t = n;

   if (i == 1)
   {
      h = n;
   }
}

cout << "\nElements in the list: \n";
link* c = h;
while (c != nullptr)
{
   cout << c->data << " ";
   c = c->next;
}
cout << endl;

cout << "Enter the element to delete: ";
cin >> ele;

link* p = nullptr;
link* d = h;
```

```
   while (d != nullptr && d->data != ele)
   {
      p = d;
      d = d->next;
   }

   if (d == h)
   {
      h = h->next;
      delete d;
   }
   else if (d != nullptr)
   {
      p->next = d->next;
      delete d;
   }

   cout << "\nElements in the list after deletion: \n";
   c = h;
   while (c != nullptr)
   {
      cout << c->data << " ";
      c = c->next;
   }

   while (h != nullptr)
   {
      link* t = h;
      h = h->next;
      delete t;
   }

   return 0;
}
```

Problem 12: Write a program to create a Circular Header Linked List of n elements and then display the list.

Answer:

```
#include <bits/stdc++.h>>
using namespace std;

struct linked_list
{
   int num;
   linked_list* next;
};

typedef linked_list node;

int main()
{
   int n, i, item;

   node* header = new linked_list;
   header->next = header;

   node* ptr = header;

   cout << "How many elements: ";
   cin >> n;

   for (i = 1; i <= n; i++)
   {
      cout << "Input a number: ";
      cin >> item;
      node* newNode = new linked_list;
      newNode->num = item;
      newNode->next = header;

      ptr->next = newNode;
      ptr = newNode;
   }
   cout << "\nElements in the Circular Header Linked List are: \n";
   ptr = header->next;
```

```
   while (ptr != header)
   {
      cout << ptr->num << endl;
      ptr = ptr->next;
   }
   ptr = header->next;
   while (ptr != header)
   {
      node* temp = ptr;
      ptr = ptr->next;
      delete temp;
   }
   delete header;

   return 0;
}
```

Problem 13: Write a program to create a Two way Linked List of n elements and then display the list.

Answer:

```
#include <bits/stdc++.h>

using namespace std;

struct linked_list
{
   int num;
   linked_list* next;
   linked_list* prev;
};

typedef linked_list node;

int main()
{
   int n, i;
   node* start, * ptr, * pre = nullptr;
```

```
start = nullptr;

cout << "How many elements: ";
cin >> n;

for (i = 1; i <= n; i++)
{
   cout << "Input a number: ";
   ptr = new linked_list;
   cin >> ptr->num;
   ptr->prev = pre;
   ptr->next = nullptr;

   if (pre != nullptr)
   {
      pre->next = ptr;
   }

   pre = ptr;

   if (i == 1)
   {
      start = ptr;
   }
}

cout << "\nElements in the Linked list are: \n";
ptr = start;
while (ptr != nullptr)
{
   cout << ptr->num << endl;
   ptr = ptr->next;
}
ptr = start;
while (ptr != nullptr)
{
```

```
        node* temp = ptr;
        ptr = ptr->next;
        delete temp;
    }

    return 0;
}
```

Problem 14: Write a program to find the 100!.

Answer:

```
#include <bits/stdc++.h>
using namespace std;

string mult(const string& a, const string& b)
{
    int len1 = a.length();
    int len2 = b.length();
    string k(len1 + len2, '0');

    for (int i = len1 - 1; i >= 0; i--)
    {
        int carry = 0;
        for (int j = len2 - 1; j >= 0; j--)
        {
            int prod = (a[i] - '0') * (b[j] - '0') + (k[i + j + 1] - '0') + carry;
            carry = prod / 10;
            k[i + j + 1] = '0' + (prod % 10);
        }
        k[i] += carry;
    }

    if (k[0] == '0')
    {
        k.erase(k.begin());
```

```
    }

    return k;
}

string fac(int n)
{
    if (n == 0 || n == 1)
    {
        return "1";
    }

    string k = "1";
    for (int i = 2; i <= n; i++)
    {
        k = mult(k, to_string(i));
    }

    return k;
}

int main()
{
    int n = 100;
    string k = fac(n);

    cout << n << "! = " << k << endl;

    return 0;
}
```

Problem 15: .Write a program to determine the value of the nth Fibonacci number Fn where Fn = Fn–1 + Fn-2 and F1 = F2 = 1 and n <= 500.

Answer:

```
#include <bits/stdc++.h>>
using namespace std;
string add(string a, string b)
```

```
{
    int len1 = a.size();
    int len2 = b.size();
    int carry = 0;
    string result;

    while (len1 > 0 || len2 > 0 || carry > 0)
    {
        int x, y;

        if (len1 > 0)
        {
            x = a[--len1] - '0';
        }
        else
        {
            x = 0;
        }

        if (len2 > 0)
        {
            y = b[--len2] - '0';
        }
        else
        {
            y = 0;
        }

        int sum = x + y + carry;
        result = char(sum % 10 + '0') + result;
        carry = sum / 10;
    }

    return result;
}

string fi(int n)
```

```
{
    string f1 = "0";
    string f2 = "1";

    if (n == 0)
    {
        return f1;
    }
    else if (n == 1)
    {
        return f2;
    }

    string f;
    for (int i = 2; i <= n; i++)
    {
        f = add(f1, f2);
        f1 = f2;
        f2 = f;
    }

    return f;
}

int main()
{
    int n = 500;
    string result = fi(n);

    cout << "F" << n << " = " << result << endl;

    return 0;
}
```