Endsem Practice

Spring 2023-24

Swapnoneel Kayal and Priyanshu Gangavati

Prof. Shivaram CS101

Question 1

Write a function “concat” which takes two 1D character arrays as input whose size is 100. You have to perform concatenation operation which appends the second string at the end of the first string without making a new character array. Display the new first string (1D character array).

Fill in the blanks in the next slide.

Eg: char A[100] = {‘h’, ‘e’, ‘l’, ‘l’, ‘o’, ‘\0’};

char B[100] = {‘w’, ‘o’, ‘r’, ‘l’, ‘d’, ‘\0’};

Now A[] becomes {‘h’, ‘e’, ‘l’, ‘l’, ‘o’,‘w’, ‘o’, ‘r’, ‘l’, ‘d’, ‘\0’}

Solution 1

```
_______ concat(char A[100], char B[100])
{
    int i = 0, j = 0;
    while(__________)
    {
        i++;
    }

    while(B[j] != __________)
    {
        _____________;
        i++;
        j++;
    }
}
```

Question 2

Given an integer array nums, find the subarray with the largest sum and return it’s sum.

Note : You only have to return the sum and not the subarray.

Input : nums = [-2, 1, -3, 4, -1, 2, 1, -5, 4]

Output : 6

Explanation : The subarray [4, -1, 2, 1] has the largest sum 6.

Solution 2 : Kadane’s Algorithm

- The main Intuition behind Kadane’s algorithm is :
    - The subarray with negative sum is discarded (by assigning max_ending_here = 0 in code).
    - We carry subarray till it gives positive sum.

- Follow the below steps to implement the idea:
    - Initialize the variables max_so_far = INT_MIN and max_ending_here = 0
    - Run a for loop from 0 to N-1 and for each index i:
        - Add the arr[i] to max_ending_here.
        - If max_so_far is less than max_ending_here then update max_so_far to max_ending_here.
        - If max_ending_here < 0 then update max_ending_here = 0
    - Return max_so_far

Question 3

You have a (2x3) 2D character array where each row contains the coefficients of a quadratic polynomial. The numbers are in character format. Given a value of x from the user, return true if the value of the polynomial of first row is greater than the value of the polynomial of second row. The coefficients in each row are in the order of x^2, x, 1. Example input is shown below.

```
char A[2][3] = {{‘1’, ‘3’, ‘6’},
                {‘2’, ‘3’, ‘8’}}
```

Thus the polynomials are (x^2 + 3x + 6) and (2x^2 + 3x + 8).

Solution 3

```
double polynomial(int a, int b, int c, double x)
{
    return a*x*x + b*x + c;
}

bool check(char A[2][3], double x)
{
    double y[2];

    int a, b, c;
    for(int i = 0; i < 2; i++)
    {
        a = A[i][0] - '0';
        b = A[i][1] - '0';
        c = A[i][2] - '0';

        y[i] = polynomial(a, b, c, x);
    }

    return (y[0] > y[1] ? true : false);
}
```

Question 4

Write a function to reverse an integer array using recursion without using loops. You have to to this in-place which means you cannot create a new array.

Fill in the blanks in the next slide.

```
void reverse(int A[], int n, int i, int &j, int &flag)
{
    int ele;
    if(_________)
    {
        ele = A[i];
        i++;
        if(i == n)
        {
            flag = 1;
        }
        else{
            _____________
        }
    }

    A[j] = ________;
    j++;
}
```

Question 4

You are given a positive integer ‘n’.

Your task is to find and return its square root. If ‘n’ is not a perfect square, then return the floor value of sqrt(n).

Example:

Input: ‘n’ = 7

Output: 2

Explanation:

The square root of the number 7 lies between 2 and 3, so the floor value is 2.

Solution 4

Brute Force Solution : Let us say, we have to find the floor value of sqrt(n)

- For this given ‘n’, we iterate ‘i’ from 1 to ‘n’. Let ans = 1.
- If (i^2 <= n), then ans ← i
- Else, break the loop and print ans.
- Time Complexity : O(n)

Binary Search Solution : Time Complexity : O(log n)

- Low = 1 ; High = n ; Mid = (Low + High)/2 ; ans = 1
- If (Mid*Mid <= n), then : Low ← Mid + 1 && ans ← mid
- Otherwise, High = Mid - 1
- Keep on reducing the search space (i.e. perform the above steps if your low is less than equal to high) and finally print ans.

Question 5

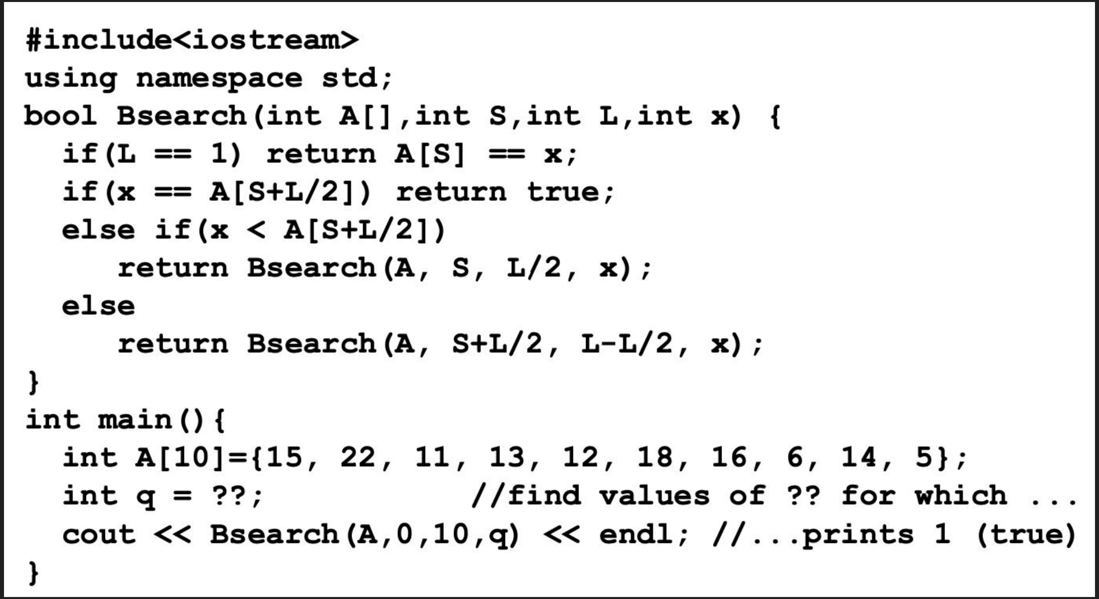
Design a structure called Shape which has 4 attributes called corners, r, l1, l2 denoting the no. of corners, radius, length of side 1 and length of side 2 respectively. Your objective to is to calculate the area of the given shape. Accept the no. of corners from the user (which can be 4 or 0 only). If the no. of corners are 0 then its a circle otherwise its a rectangle (possibly a square).

```cpp
struct Shape
{
    public:
    int corners;
    double r, l1, l2;

    Shape()
    {
        corners = 0;
        r = 0.0;
        l1 = 0.0;
        l2 = 0.0;
    }

    double circle()
    {
        return 3.14159*r*r;
    }

    double rectangle()
    {
        return l1*l2;
    }
};
```

```
int main()
{
    Shape s;

    int corner;
    cin>>corner;

    s.corners = corner;
    if(s.corners == 0)
    {
        cin>>s.r;
        cout<<s.circle()<<endl;
    }
    else{
        cin>>s.l1>>s.l2;
        cout<<s.rectangle()<<endl;
    }

    return 0;
}
```

Question 6

The function BSearch (binary search) shown below, expects the array 'A' to be sorted in increasing order, and guarantees that if the element ‘x’ to be searched is present in 'A', ‘true’ is returned; else `false’ is returned. Suppose 'A' is not sorted (in increasing order) at all, and A={15,22,11,13,12,18,16,6,14,5}. For what values of ‘??’ will Bsearch successfully find the index of ‘q’ in 'A' and return “true” (i.e., 1)?



```
#include<iostream>
using namespace std;
bool Bsearch(int A[],int S,int L,int x) {
  if(L == 1) return A[S] == x;
  if(x == A[S+L/2]) return true;
  else if(x < A[S+L/2])
     return Bsearch(A, S, L/2, x);
  else
     return Bsearch(A, S+L/2, L-L/2, x);
}
int main(){
  int A[10]={15, 22, 11, 13, 12, 18, 16, 6, 14, 5};
  int q = ??;           //find values of ?? for which ...
  cout << Bsearch(A,0,10,q) << endl; //...prints 1 (true)
}
```

Solution 6

First it checks for x (> or <=) A[5] => x= A[5] =>X = 18 will return true

Now Bsearch will search in the first half of the array if x < 18, else in the second half, but we can see that the second half has values less than 18.

=>Hence X can be none of those values.

Next search goes to index int(5/2) = 2 , we see that x = A[2] < 18.

So x = 11 will return True.

Now to match with the first quarter of the array, x < 11, which we see isn't possible

=>Hence X can be none of those values.

So, can only match in the 2nd quarter of the array, A[2 + 3/2] =A[3] = 13

11< 13 < 18, hence returns True. The only other element A[4] is less than 13 but is to the right of 13, therefore cannot be matched.

So the only values of q that return true is 11,13,18.