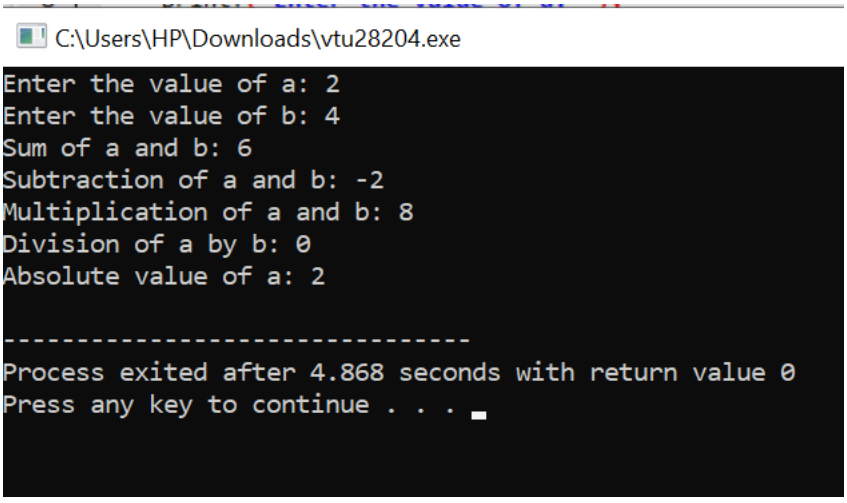
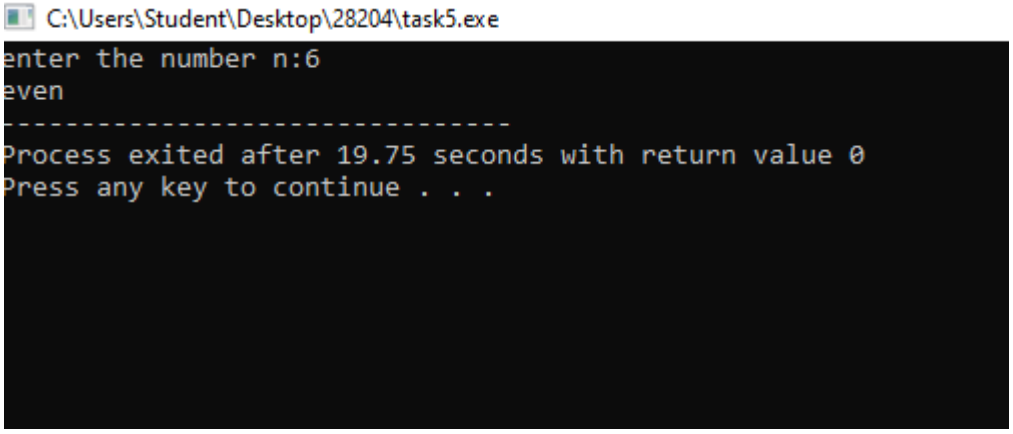
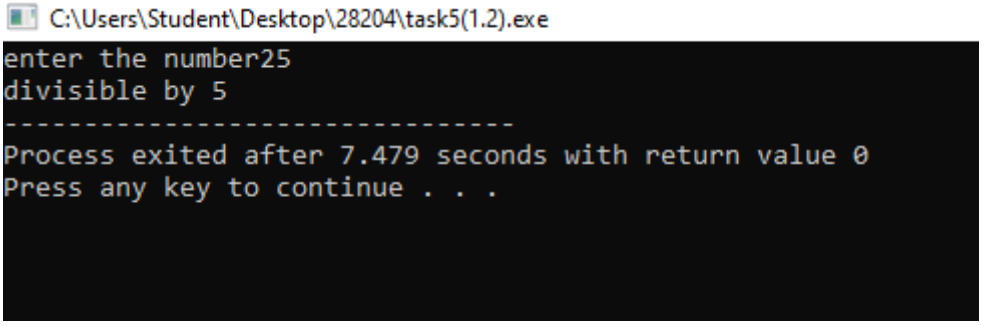
```
#include<stdlib.h>

int main() {

int a,b;

printf("enter the values of a:");

scanf("%d" , &a);

printf("enter the value of b:");

scanf("%d",&b);

printf("sum of a and b:%d/n,a+b");

printf("subtraction of a and b:%d/n,a-b");

printf("multiplcation of a and b:%d/n",a*b);

printf("divison of a and b:%d/n",a/b);

printf("absolute value of a:%d/n",abs(a));

return 0;

}
```



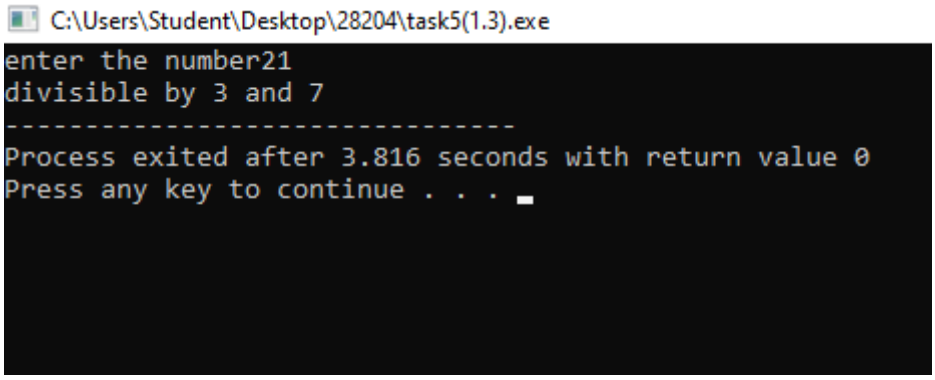
```
#include<stdio.h>

int main () {

int n;

printf("enter the number n:");

scanf("%d", &n);

if(n%2==0) {

printf("even");

}else{

printf("odd");

}

return 0;

}
```



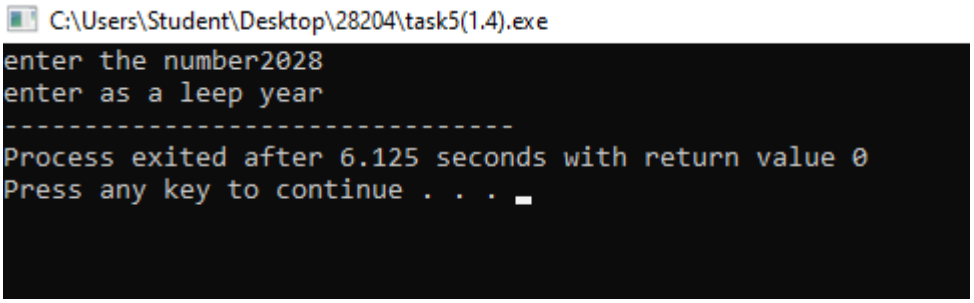
```
#include<stdio.h>

int main () {

int n;

printf("enter the number");

scanf("%d", &n);

if(n%5==0){

printf("divisible by 5");

}

else{

printf("not divisible by 5");

}

}
```



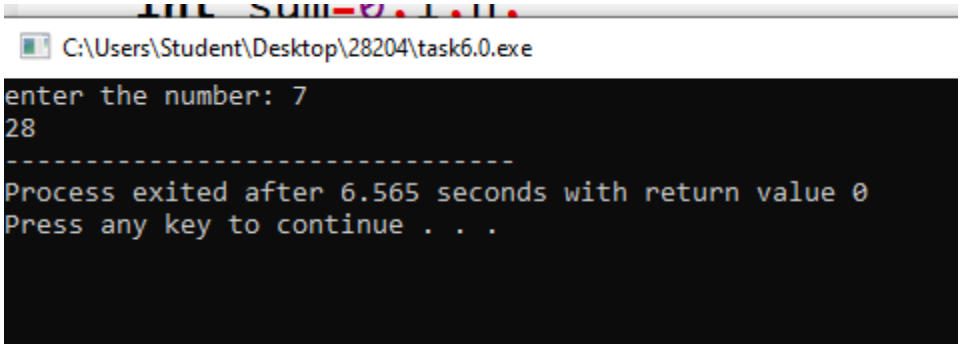
```
#include<stdio.h>
int main() {
int n;
printf("enter the number");
scanf("%d" ,&n);
if(n%7==0&&n%3==0){
printf("divisible by 3 and 7");
}else{
printf("divisible by 3 and 7");
}
return 0;
}
```



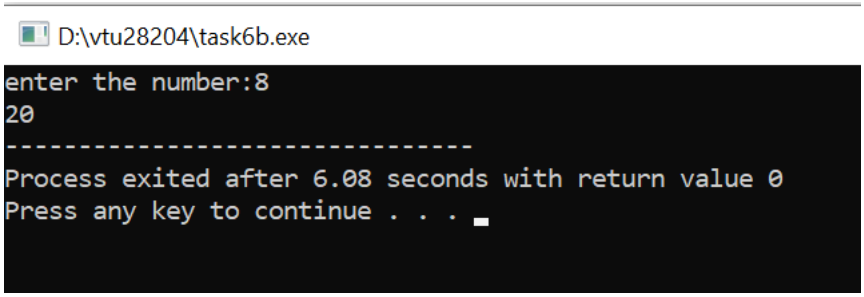
```
#include<stdio.h>

int main() {

int n;

printf("enter the number");

scanf("%d" ,&n);

if(n%4==0){

printf("enter as a leep year");

}else{

printf("enter as not a leep year");

}

return 0;

}
```



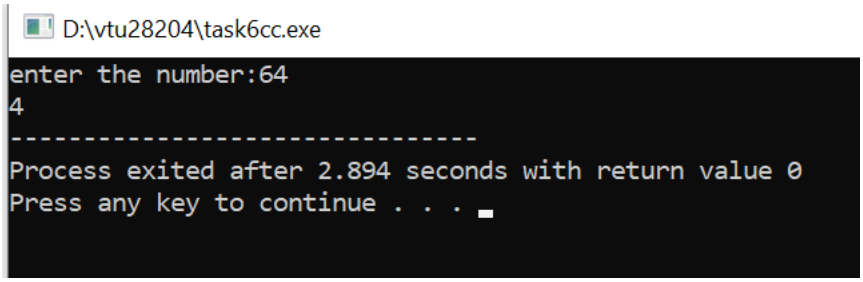
```
#include<stdio.h>
int main(){
int sum=0,i,n;
printf("enter the number: ");
scanf("%d",&n);
for(i=0;i<=n;i++){
sum=sum+i;
}
printf("%d",sum);
}
```



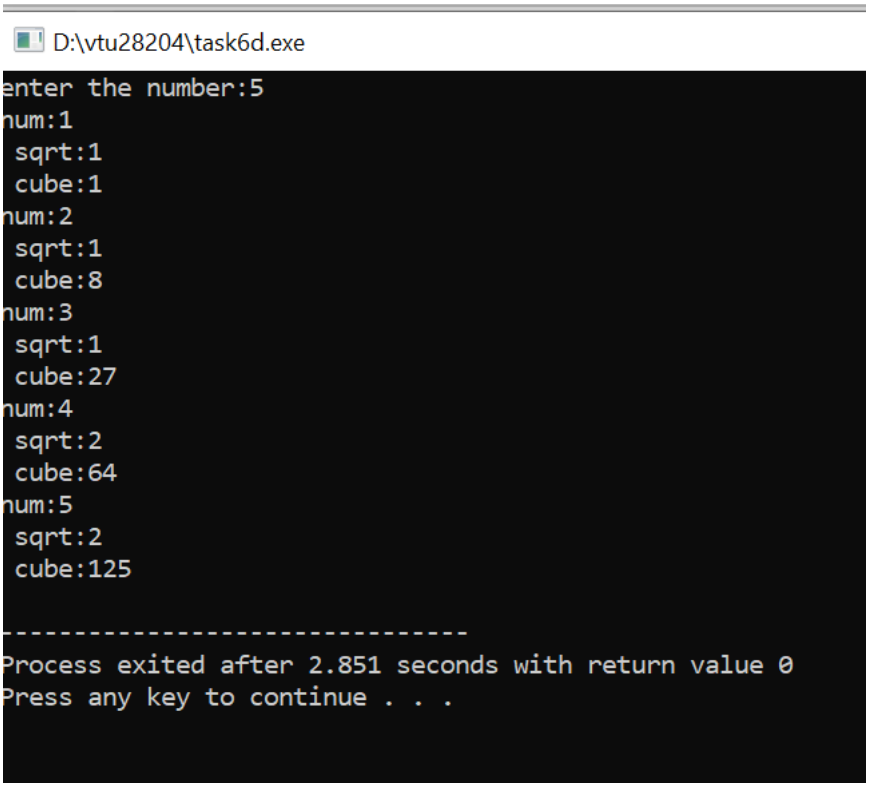
```
#include<stdio.h>
int main(){
int sum=0,i,n;
printf("enter the number:");
scanf("%d",&n);
for(i=1;i<=n;i++){
if (i%2==0){
sum=sum+i;
}
}
printf("%d",sum);
return 0;
}
```



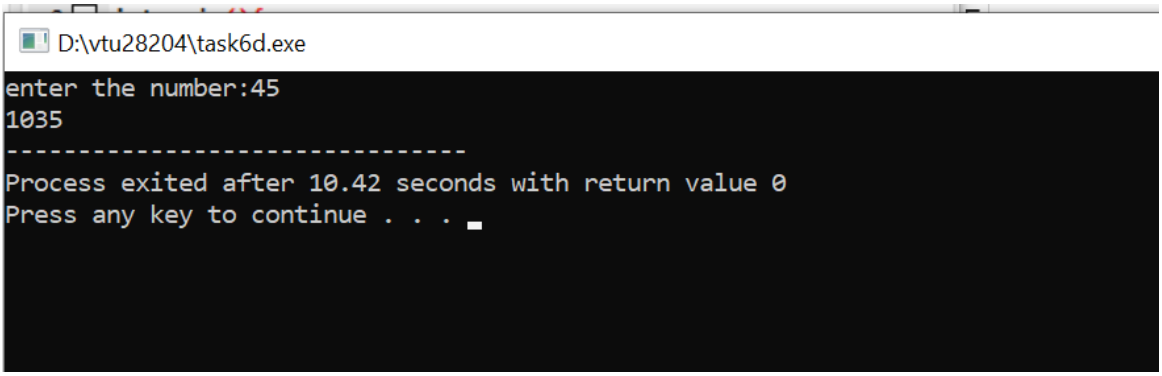
```c
#include<stdio.h>
#include<math.h>
int main(){
	int result,n;
	printf("enter the number:");
	scanf("%d",&n);
	result=cbrt(n);
	printf("%d",result);
	return 0;
}
```



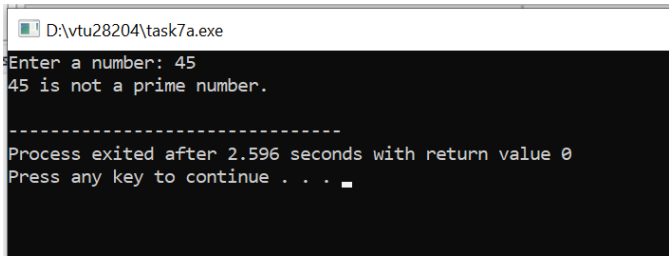
```
#include<stdio.h>
#include<math.h>
int main(){
int result1,result2,i,n;
printf("enter the number:");
scanf("%d",&n);
for(i=1;i<=n;i++){
result1=sqrt(i);
result2=i*i*i;
printf("num:%d\n sqrt:%d\n cube:%d\n",i,result1,result2);
}
return 0;
}
```



```
#include<stdio.h>
int main(){
int sum=0,i,n;
printf("enter the number:");
scanf("%d",&n);
for(i=1;i<=n;i++){
sum=sum+i ;
}
printf("%d",sum);
return 0;
}
```

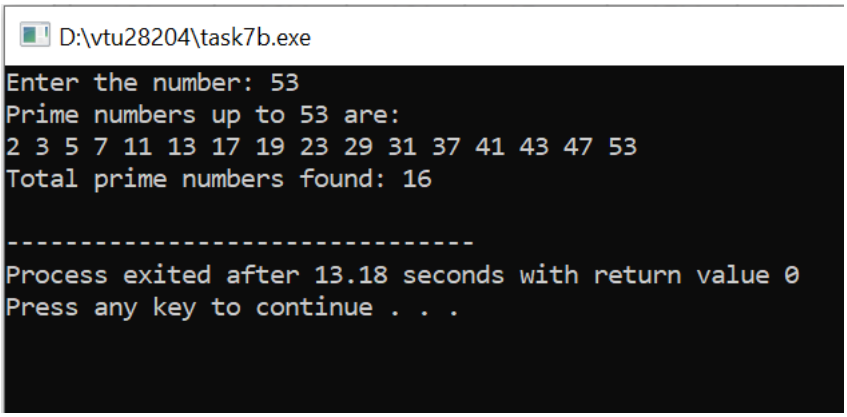


```
#include <stdio.h>

int main() {

int num, i, isPrime = 1; // assume primeprintf("Enter a number: ");

scanf("%d", &num);

if (num <= 1) {

isPrime = 0; // 0 and 1 are not prime

} else {

for (i = 2; i * i <= num; i++) {

if (num % i == 0) {

isPrime = 0;

break;

}

}

}

if (isPrime)

printf("%d is a prime number.\n", num);

else

printf("%d is not a prime number.\n", num);

return 0;}
```



```c
#include <stdio.h>
int isPrime(int num) {
    int i; // Declare i outside the loop
    if (num <= 1) {
        return 0;
    }
    for (i = 2; i * i <= num; i++) {
        if (num % i == 0) {
            return 0;
        }
    }
    return 1;
}

int main() {
    int n, count = 0, i; // Declare i here for the main loop

    printf("Enter the number: ");
    scanf("%d", &n);

    if (n < 2) {
        printf("There are no prime numbers less than %d.\n", n);
        return 0;
    }

    printf("Prime numbers up to %d are:\n", n);
    for (i = 2; i <= n; i++) {
        if (isPrime(i)) {
```

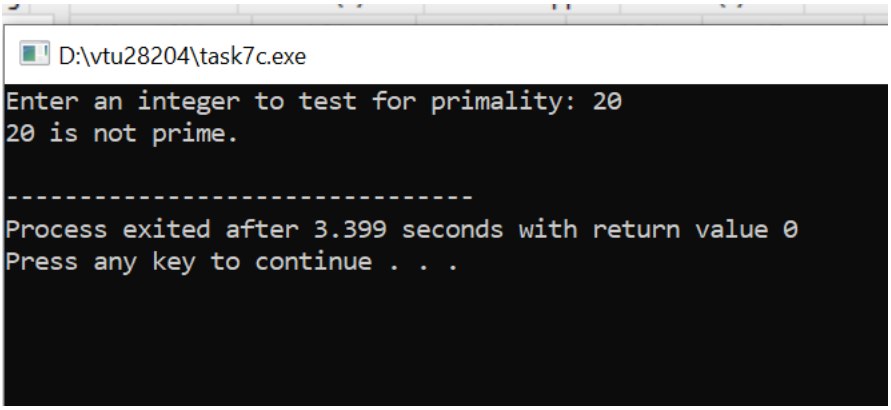
```
            printf("%d ", i);
            count++;
        }
    }

    printf("\nTotal prime numbers found: %d\n", count);

    return 0;
}
```



```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define ll long long

// Function to compute (base^exponent) % mod efficiently
ll modulo(ll base, ll exponent, ll mod) {
    ll result = 1;
    base = base % mod;

    while (exponent > 0) {
        if (exponent % 2 == 1) {
            result = (result * base) % mod;
        }
        base = (base * base) % mod;
        exponent = exponent / 2;
    }

    return result;
}

// Fermat primality test function
int Fermat(ll p, int iterations) {
    int i;
    ll a;

    if (p <= 1) return 0;
```

```
    for (i = 0; i < iterations; i++) {
        a = rand() % (p - 1) + 1; // Random a in range [1, p-1]
        if (modulo(a, p - 1, p) != 1) {
            return 0; // Definitely composite
        }
    }

    return 1; // Probably prime
}

int main() {
    int iterations = 50;
    ll num;

    // Seed random number generator
    srand(time(NULL));

    // Input
    printf("Enter an integer to test for primality: ");
    scanf("%lld", &num);

    // Test and output
    if (Fermat(num, iterations))
        printf("%lld is probably prime.\n", num);
    else
        printf("%lld is not prime.\n", num);
```

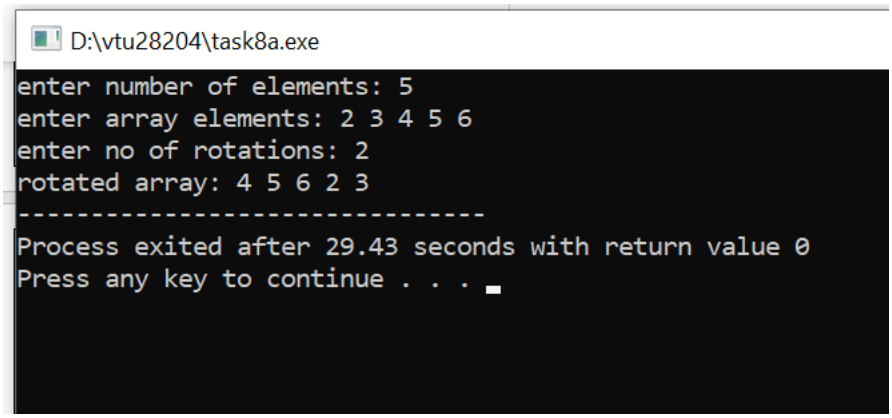
```
    return 0;
}
```


D:\vtu28204\task7c.exe
Enter an integer to test for primality: 20
20 is not prime.

Process exited after 3.399 seconds with return value 0
Press any key to continue . . .

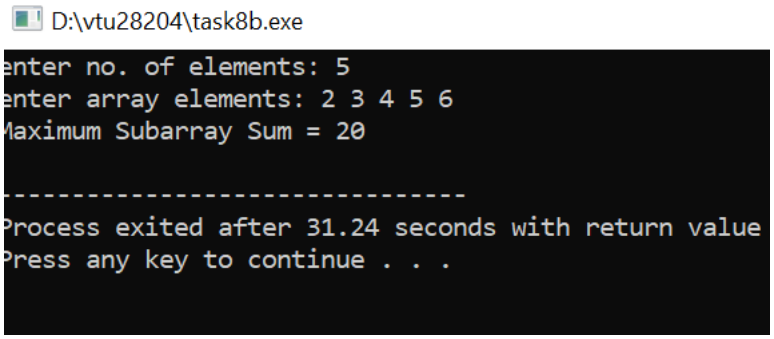
```c
#include <stdio.h>

int main() {
    int n, d;
    int i, j; // ? Declare loop variables here
    int arr[100];

    printf("enter number of elements: ");
    scanf("%d", &n);

    printf("enter array elements: ");
    for (i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }

    printf("enter no of rotations: ");
    scanf("%d", &d);

    d = d % n;

    for (i = 0; i < d; i++) {
        int first = arr[0]; // ? Still OK here (C allows it inside loop body)
        for (j = 0; j < n - 1; j++) {
            arr[j] = arr[j + 1];
        }
        arr[n - 1] = first;
    }
```

```
    printf("rotated array: ");
    for (i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }

    return 0;
}
```



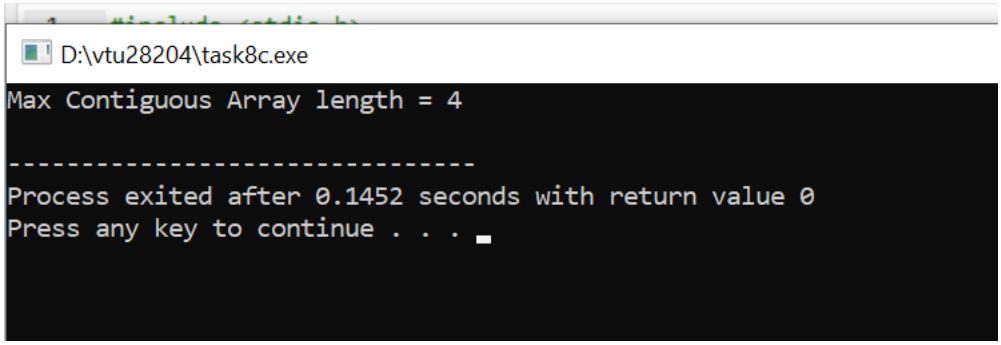
```c
#include <stdio.h>

int main() {
    int n, i;

    printf("enter no. of elements: ");
    scanf("%d", &n);

    int arr[100];

    printf("enter array elements: ");
    for (i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }

    int max_sum = arr[0];
    int current_sum = arr[0];

    for (i = 1; i < n; i++) {  // ? Use existing declared i
        if (current_sum < 0) {
            current_sum = arr[i];
        } else {
            current_sum += arr[i];
        }

        if (current_sum > max_sum) {
            max_sum = current_sum;
```

```
        }
    }

    printf("Maximum Subarray Sum = %d\n", max_sum);

    return 0;
}
```



```c
#include <stdio.h>
#include <string.h>

int maxlength(int nums[], int n) {
    int sum = 0, max_len = 0;
    int hash[2 * 100 + 1];  // Assuming max n is 100
    int i;

    for (i = 0; i < 2 * n + 1; i++) {
        hash[i] = -2;
    }

    hash[n] = -1;

    for (i = 0; i < n; i++) {
        sum += (nums[i] == 0 ? -1 : 1);

        if (hash[sum + n] != -2) {
            if (i - hash[sum + n] > max_len) {
                max_len = i - hash[sum + n];
            }
        } else {
            hash[sum + n] = i;
        }
    }

    return max_len;
}
```

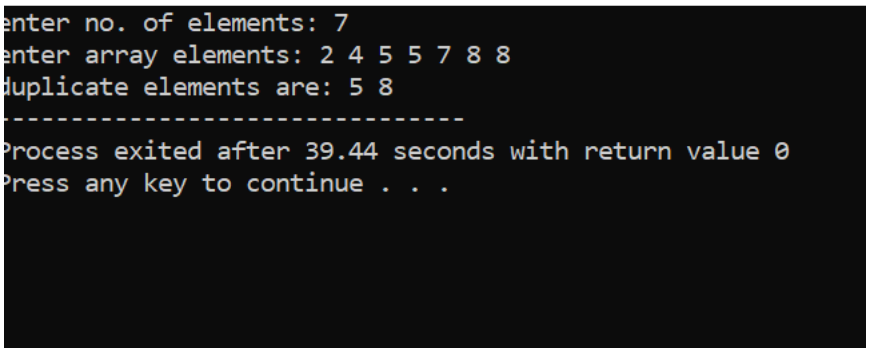
```
int main() {
    int nums[] = {0, 1, 0, 1, 0, 0};
    int n = sizeof(nums) / sizeof(nums[0]);

    printf("Max Contiguous Array length = %d\n", maxlength(nums, n));

    return 0;
}
```



```c
#include <stdio.h>

int main() {
    int n, i, j;
    int arr[100];
    int found = 0;

    printf("enter no. of elements: ");
    scanf("%d", &n);

    printf("enter array elements: ");
    for (i = 0; i < n; i++)
        scanf("%d", &arr[i]);

    printf("duplicate elements are: ");

    for (i = 0; i < n; i++) {
        for (j = i + 1; j < n; j++) {
            if (arr[i] == arr[j]) {
                printf("%d ", arr[j]);
                found = 1;
                break; // To avoid printing the same duplicate multiple times
            }
        }
    }

    if (!found)
        printf("no duplicates found");
```

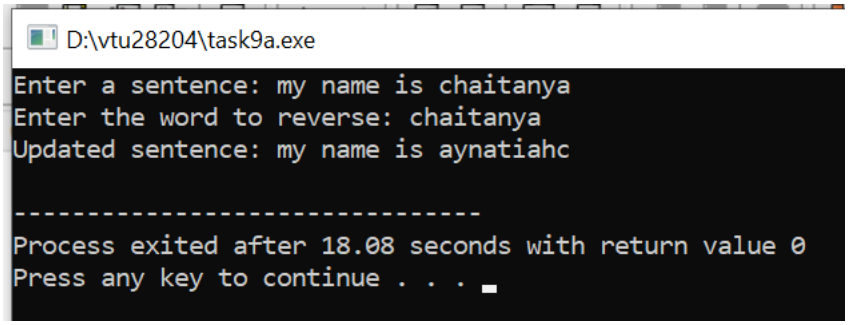
```
    return 0;
}
```

D:\vtu28204\task8d.exe



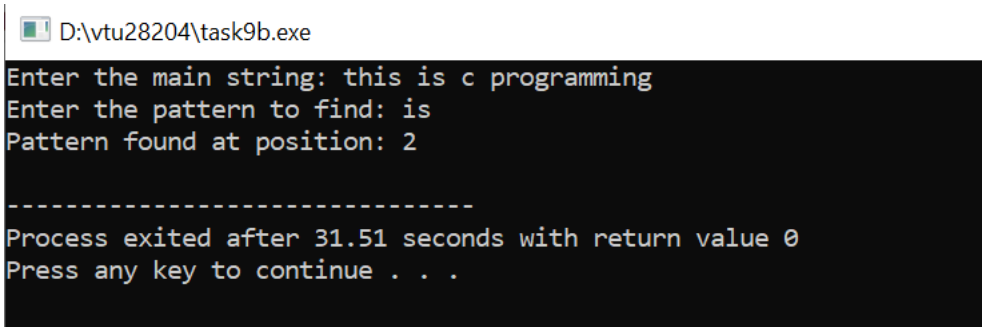
```c
#include <stdio.h>
#include <string.h>

int main() {
    char str[200], word[50];
    int i;  // declare loop variable here
    printf("Enter a sentence: ");
    scanf("%[^\n]s", str); // read full line with spaces

    printf("Enter the word to reverse: ");
    scanf(" %s", word); // space before %s to ignore leftover newline

    int len = strlen(str);
    int wlen = strlen(word);

    for (i = 0; i <= len - wlen; i++) {
        // Check if the word matches at position i
        if (strncmp(&str[i], word, wlen) == 0) {
            // Ensure it's a complete word (check boundaries)
            if ((i == 0 || str[i - 1] == ' ') && (str[i + wlen] == ' ' || str[i + wlen] == '\0')) {
                int start = i;
                int end = i + wlen - 1;

                while (start < end) {
                    char temp = str[start];
                    str[start] = str[end];
                    str[end] = temp;
                    start++;
```

```
            end--;
        }
        break; // reverse only the first occurrence
      }
    }
  }

  printf("Updated sentence: %s\n", str);
  return 0;
}
```

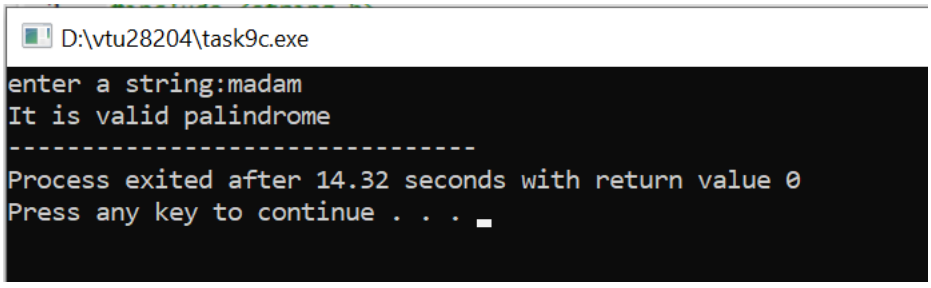


```c
#include <stdio.h>
#include <string.h>

int main() {
    char str[200], pattern[50];
    int found = 0;
    int i, j;  // Declare loop variables here

    printf("Enter the main string: ");
    scanf("%[^\n]s", str);

    printf("Enter the pattern to find: ");
    scanf(" %s", pattern);

    int lenStr = strlen(str);
    int lenPat = strlen(pattern);

    for (i = 0; i <= lenStr - lenPat; i++) {
        for (j = 0; j < lenPat; j++) {
            if (str[i + j] != pattern[j]) {
                break;
            }
        }

        // If full pattern matched
        if (j == lenPat) {
            printf("Pattern found at position: %d\n", i);
            found = 1;
```

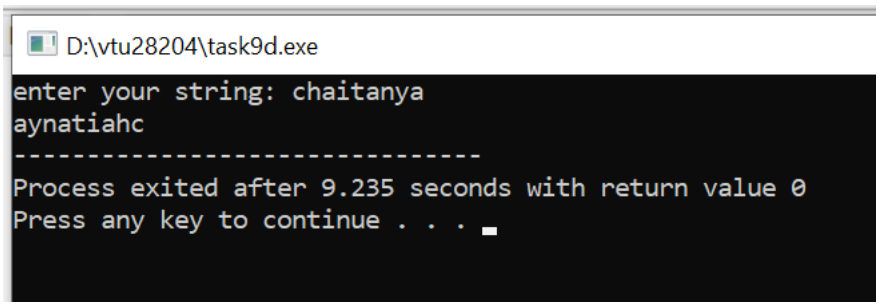
```
            break;  // Remove this line if you want to find all occurrences
        }
    }

    if (!found)
        printf("Pattern not found!\n");

    return 0;
}
```



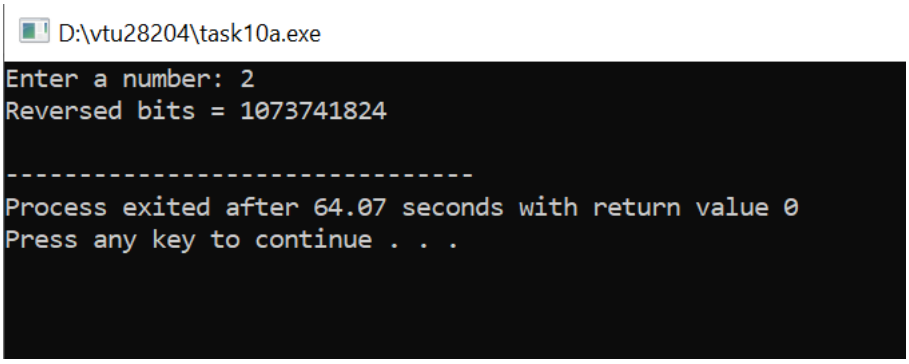
```c
#include <stdio.h>
#include <string.h>
int main() {
   char str[100];
   int i, len, ispalindrome = 1;
   printf("enter a string:");
   scanf("%s", &str);
   len = strlen(str);
   for (i = 0; i < len / 2; i++) {
      if (str[i] != str[len - i - 1]) {
         ispalindrome = 0;
         break;
      }
   }
   if (ispalindrome) {
      printf("It is valid palindrome");
   } else {
      printf("It is not valid palindrome");
   }
   return 0;
}
```



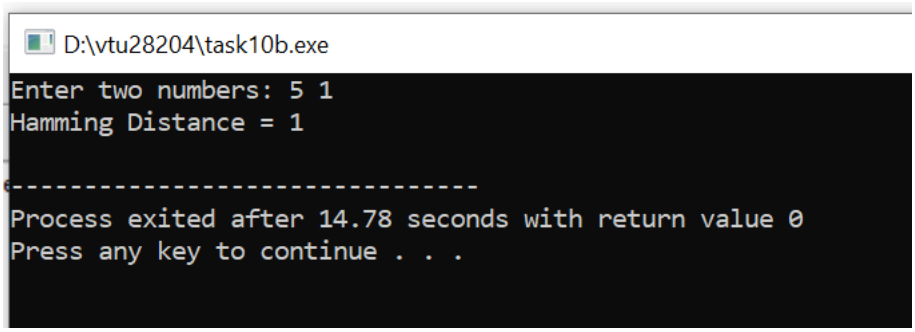
```c
#include <stdio.h>
#include <string.h>

int main() {
    char str[100];
    int i;

    printf("enter your string: ");
    scanf("%s", str);  // No need for & when reading a string

    int len = strlen(str);

    for (i = len - 1; i >= 0; i--) {
        printf("%c", str[i]);
    }

    return 0;
}
```



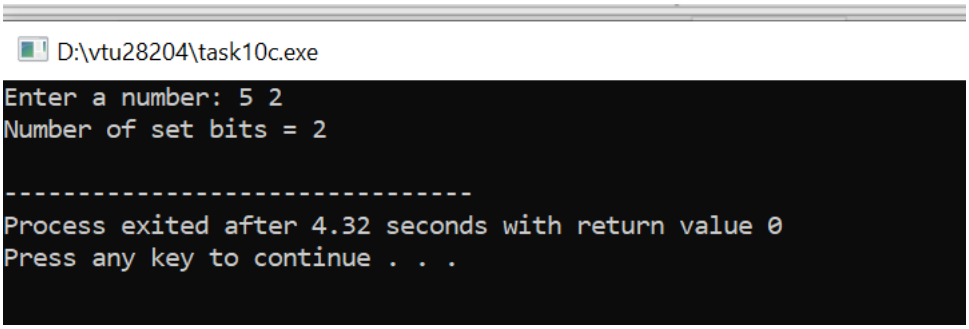
```c
#include <stdio.h>
int main() {
   unsigned int num, rev = 0;
   int i;
   printf("Enter a number: ");
   scanf("%u", &num);
   for (i = 0; i < 32; i++) {
      rev <<= 1;
      if (num & 1)
         rev |= 1;
      num >>= 1;
   }
    printf("Reversed bits = %u\n", rev);
   return 0;
}
```



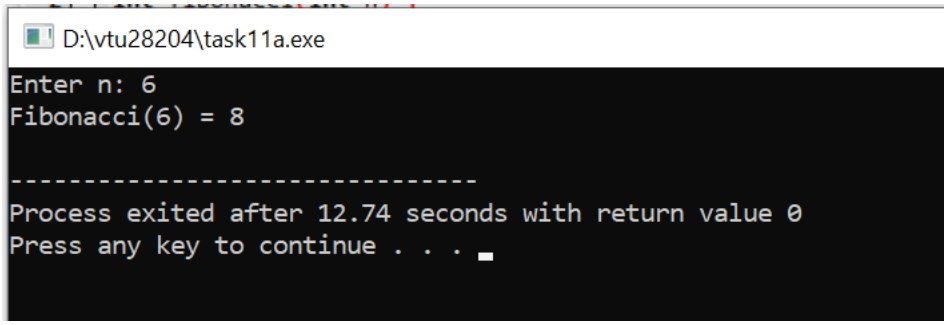
```c
#include <stdio.h>
int main() {
    unsigned int a, b, xor_result;
    int count = 0;
    printf("Enter two numbers: ");
    scanf("%u %u", &a, &b);
    xor_result = a ^ b;
    while (xor_result != 0) {
        if (xor_result & 1)
            count++;
        xor_result >>= 1;
    }
    printf("Hamming Distance = %d\n", count);
    return 0;
}
```



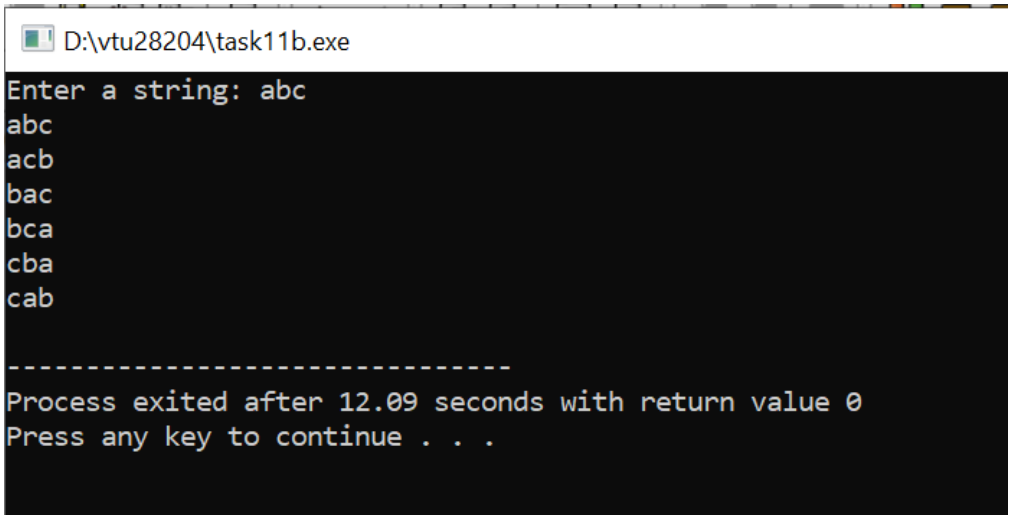
```c
#include <stdio.h>
int main() {
   unsigned int num;
   int count = 0;
   printf("Enter a number: ");
   scanf("%u", &num);
   while (num != 0) {
      if (num & 1)
         count++;
      num >>= 1;
   }
   printf("Number of set bits = %d\n", count);
   return 0;
}
```



```c
#include <stdio.h>
int fibonacci(int n) {
    if (n <= 1)
        return n;
    return fibonacci(n-1) + fibonacci(n-2);
}
int main() {
    int n;
    printf("Enter n: ");
    scanf("%d", &n);
    printf("Fibonacci(%d) = %d\n", n, fibonacci(n));
    return 0;
}
```

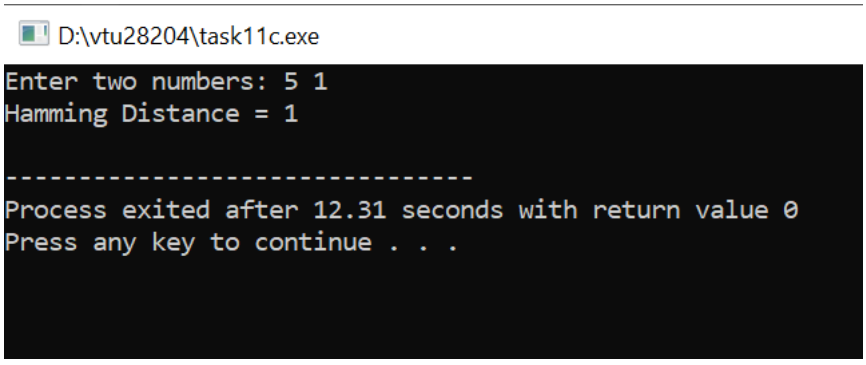


```c
#include <stdio.h>
#include <string.h>
void swap(char *x, char *y) {
   char temp = *x;
   *x = *y;
   *y = temp;
}
void permute(char *str, int l, int r) {
          int i;
   if (l == r)
      printf("%s\n", str);
   else {
      for (i = l; i <= r; i++) {
         swap(&str[l], &str[i]);
         permute(str, l + 1, r);
         swap(&str[l], &str[i]); // backtrack
      }
   }
}
int main() {
   char str[20];
   printf("Enter a string: ");
   scanf("%s", str);
   int n = strlen(str);
   permute(str, 0, n - 1);
   return 0;
}
```


D:\vtu28204\task11b.exe
Enter a string: abc
abc
acb
bac
bca
cba
cab

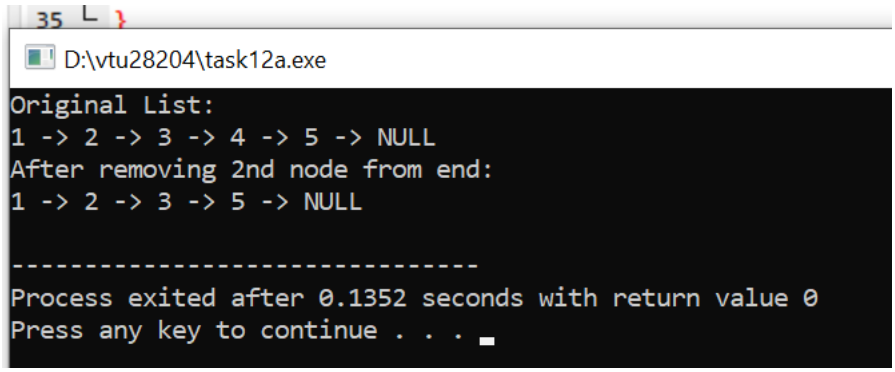
Process exited after 12.09 seconds with return value 0
Press any key to continue . . .

```c
#include <stdio.h>
int hammingDistance(int a, int b) {
    if (a == 0 && b == 0)
        return 0;
    return ( (a & 1) != (b & 1) ) + hammingDistance(a >> 1, b >> 1);
}
int main() {
    int a, b;
    printf("Enter two numbers: ");
    scanf("%d %d", &a, &b);
    printf("Hamming Distance = %d\n", hammingDistance(a, b));
    return 0;
}
```



```
#include <stdio.h>
#include <stdlib.h>
struct Node {
   int data;
   struct Node* next;
};
struct Node* createNode(int data) {
   struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
   newNode->data = data;
   newNode->next = NULL;
   return newNode;
}
void removeNthFromEnd(struct Node** head, int n) {
        int i;
   struct Node* temp = *head;
   int len = 0;
   while (temp) {
      len++;
      temp = temp->next;
   }
   if (n > len) return;
   int pos = len - n;
   temp = *head;
   if (pos == 0) {
      *head = (*head)->next;
      free(temp);
      return;
   }
```

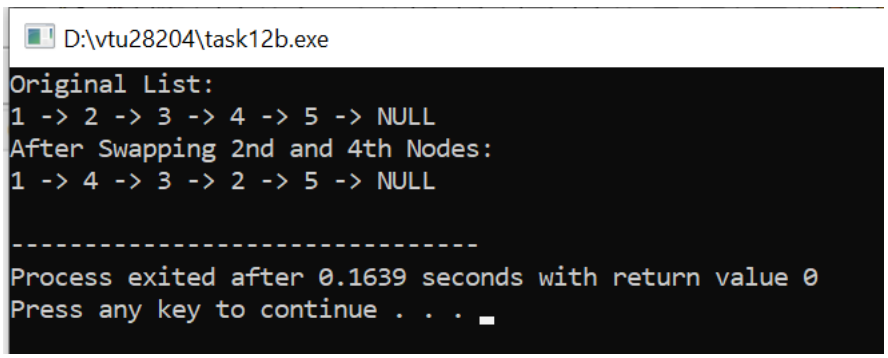
```
    for ( i = 0; i < pos - 1; i++) {
        temp = temp->next;
    }
    struct Node* delTemp = temp->next;
    temp->next = delTemp->next;
    free(delTemp);
}
void printList(struct Node* head) {
    while (head) {
        printf("%d -> ", head->data);
        head = head->next;
    }
    printf("NULL\n");
}
int main() {
    struct Node* head = createNode(1);
    head->next = createNode(2);
    head->next->next = createNode(3);
    head->next->next->next = createNode(4);
    head->next->next->next->next = createNode(5);
    printf("Original List:\n");
    printList(head);
    removeNthFromEnd(&head, 2);
    printf("After removing 2nd node from end:\n");
    printList(head);
     return 0;
}
```


35 }
D:\vtu28204\task12a.exe
Original List:
1 -> 2 -> 3 -> 4 -> 5 -> NULL
After removing 2nd node from end:
1 -> 2 -> 3 -> 5 -> NULL

Process exited after 0.1352 seconds with return value 0
Press any key to continue . . .

```
#include <stdio.h>
#include <stdlib.h>
struct Node {
   int data;
   struct Node* next;
};
struct Node* createNode(int data) {
   struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
   newNode->data = data;
   newNode->next = NULL;
   return newNode;
}
void swapNodes(struct Node* head, int x, int y) {
          int i;
   if (x == y) return;

   struct Node *a = head, *b = head;
   for (i = 1; a && i < x; i++) a = a->next;
   for ( i = 1; b && i < y; i++) b = b->next;

   if (a && b) {
      int temp = a->data;
      a->data = b->data;
      b->data = temp;
   }
}
void printList(struct Node* head) {
   while (head) {
```

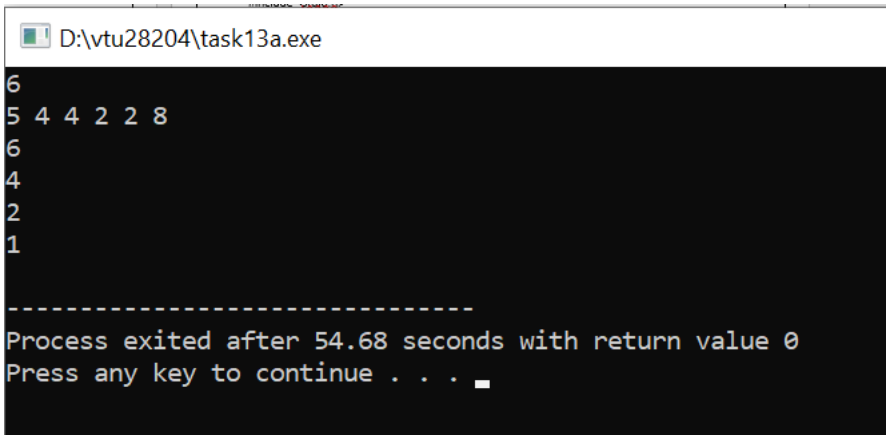
```
        printf("%d -> ", head->data);
        head = head->next;
    }
    printf("NULL\n");
}

int main() {
    struct Node* head = createNode(1);
    head->next = createNode(2);
    head->next->next = createNode(3);
    head->next->next->next = createNode(4);
    head->next->next->next->next = createNode(5);
    printf("Original List:\n");
    printList(head);
    swapNodes(head, 2, 4);
    printf("After Swapping 2nd and 4th Nodes:\n");
    printList(head);
    return 0;
}
```



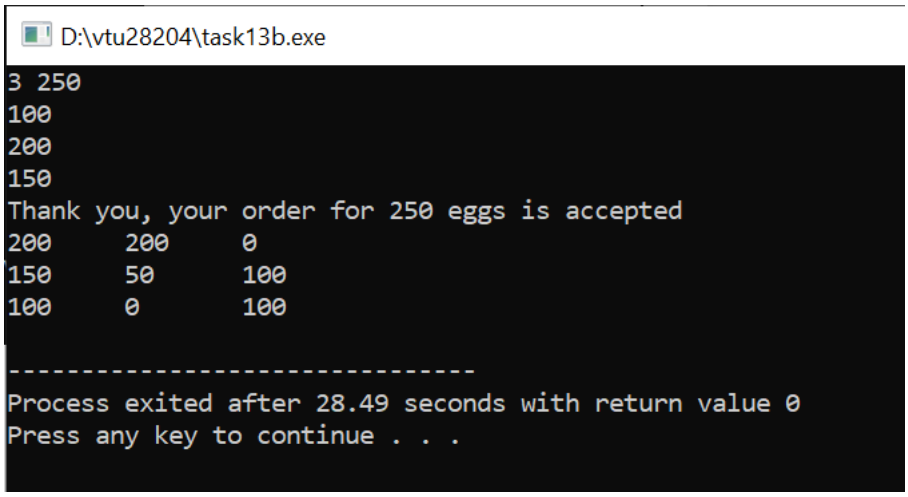
```c
#include <stdio.h>
int removeDuplicates(int arr[], int n) {
        int i;
   if (n == 0) return 0;
   int j = 0;
   for ( i = 1; i < n; i++) {
      if (arr[i] != arr[j]) {
         j++;
         arr[j] = arr[i];
      }
   }
   return j + 1;
}
int main() {
        int i;
   int arr[] = {1, 1, 2, 2, 3, 4, 4, 5};
   int n = sizeof(arr) / sizeof(arr[0]);
   int newLen = removeDuplicates(arr, n);
   printf("Array after removing duplicates:\n");
   for ( i = 0; i < newLen; i++)
      printf("%d ", arr[i]); return 0;}
```

```
D:\vtu28204\task12c.exe
Array after removing duplicates:
1 2 3 4 5
--------------------------------
Process exited after 0.1653 seconds with return value 0
Press any key to continue . . .
```

```
#include <stdio.h>
int main() {
        int i;
   int n,a[n];
   scanf("%d", &n);
   for(i=0;i<n;i++) scanf("%d",&a[i]);
   while(1){
        int i;
      int min=9999, count=0;
      for( i=0;i<n;i++)
         if(a[i]>0 && a[i]<min) min=a[i];
      for( i=0;i<n;i++)
         if(a[i]>0){ a[i]-=min; count++; }
      if(count==0) break;
      printf("%d\n",count);
   }
   return 0;
}
```



```c
#include <stdio.h>
int main(){
        int i,j;
   int M, X; scanf("%d%d",&M,&X);
   int eggs[M], alloc[M], bal[M], total=0;
   for( i=0;i<M;i++){ scanf("%d",&eggs[i]); total+=eggs[i]; }

   int order = (X>=total)? total-1 : X;
   if(X>=total) printf("Sorry, we can only supply %d eggs\n",order);
   else printf("Thank you, your order for %d eggs is accepted\n",order);

   // Sort indices by descending order (simple bubble)
   for( i=0;i<M-1;i++)
      for(j=i+1;j<M;j++)
         if(eggs[j]>eggs[i]){
            int t=eggs[i]; eggs[i]=eggs[j]; eggs[j]=t;
         }

   for( i=0;i<M;i++){
      if(order>0){
         alloc[i]=(order>=eggs[i])?eggs[i]:order;
         order-=alloc[i];
      } else alloc[i]=0;
      bal[i]=eggs[i]-alloc[i];
      printf("%d\t%d\t%d\n",eggs[i],alloc[i],bal[i]);
   }
   return 0;
}
```

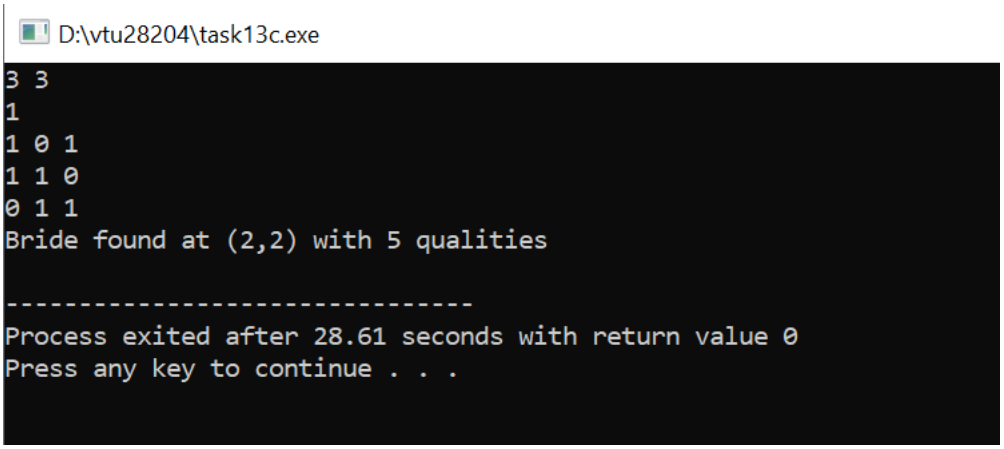

D:\vtu28204\task13b.exe
3 250
100
200
150
Thank you, your order for 250 eggs is accepted
200 200 0
150 50 100
100 0 100

Process exited after 28.49 seconds with return value 0
Press any key to continue . . .

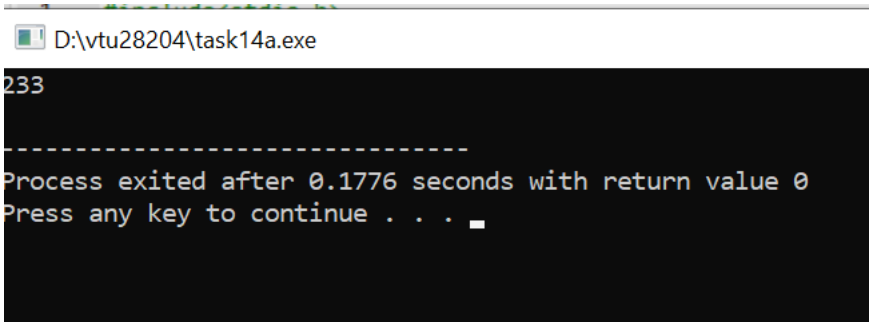
```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

int main() {
   int n, m;
   int i, j, x, y;
   int r = -1, c = -1, tie = 0;
   int maxQ = 0, minDist = 999;

   scanf("%d%d", &n, &m);
   int a[n][m];  // Variable-length array (VLA) is OK in C99+, may fail in Turbo C

   // Input
   for (i = 0; i < n; i++)
      for (j = 0; j < m; j++)
         scanf("%d", &a[i][j]);

   // Processing
   for (i = 0; i < n; i++) {
      for (j = 0; j < m; j++) {
         if (a[i][j] == 1 && !(i == 0 && j == 0)) {
            int q = 0;
            for (x = -1; x <= 1; x++) {
               for (y = -1; y <= 1; y++) {
                  int ni = i + x;
                  int nj = j + y;
                  if (ni >= 0 && nj >= 0 && ni < n && nj < m && !(x == 0 && y == 0)) {
```

```
                    if (a[ni][nj] != 0) q++;
                }
            }
        }
        int dist = abs(i - 0) + abs(j - 0);
        if (q > maxQ) {
            maxQ = q;
            minDist = dist;
            r = i;
            c = j;
            tie = 0;
        } else if (q == maxQ) {
            if (dist < minDist) {
                minDist = dist;
                r = i;
                c = j;
                tie = 0;
            } else if (dist == minDist) {
                tie = 1;
            }
        }
    }
}

// Output
if (r == -1)
    printf("No suitable girl found\n");
```

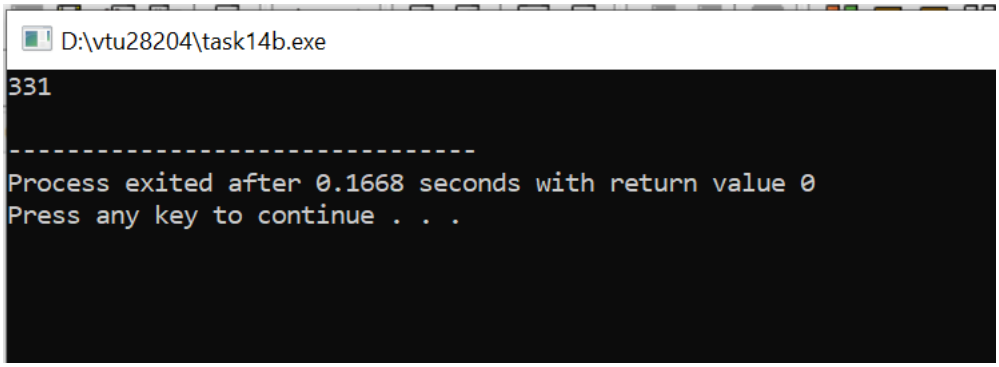
```
    else if (tie)
        printf("Polygamy not allowed\n");
    else
        printf("Bride found at (%d,%d) with %d qualities\n", r + 1, c + 1, maxQ);

    return 0;
}
```



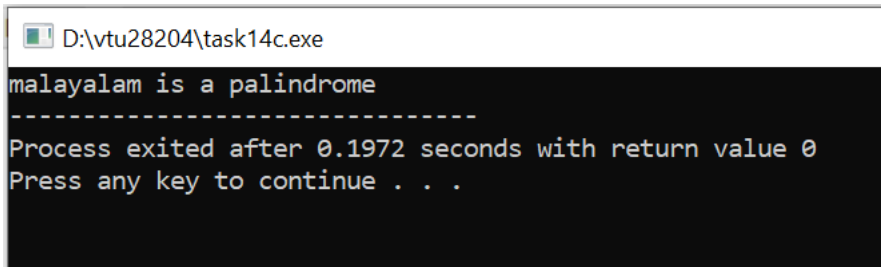
```c
#include<stdio.h>
int main()
{
   int x = 4, y, z;
   y = --x;
   z = x--;
   printf("%d%d%d\n", x, y, z);
   return 0;
}
```



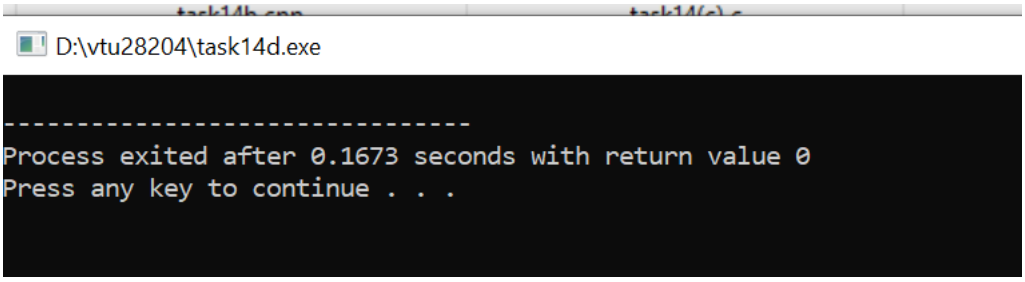
```c
#include<stdio.h>
int main(){
   int x = 4, y = 3, z;
   z = x-- - y;
   printf("%d%d%d\n", x, y, z);
   return 0;}
```



```c
#include<stdio.h>
int main(){
   while('a' < 'b'){
      printf("malayalam is a palindrome");
      return 0;
   }
}
```



```c
#include<stdio.h>
int main(){
   int i = 0;
   for(; i;)
      printf("here is some mail for you \n");
   return 0;
}
```



```
#include<stdio.h>
int main(){
   float x = 1.1;
   while(x == 1.1) {
      printf("%f\n", x);
      x = x - 0.1;
   }
   return 0;}
```