C Set 2

**1)**

#include <stdio.h>

int Main()

{

int n i;

printf(Enter an integer: );

scanf("%d",&n);

for(i=1; i<=10; ++i)

{

printf("%d \* %d = %d \n", n, i, n\*i);

return 0

}

Ans:

(i) int main()

(ii) int n, i ;

(iii) parenthesis in printf(“….”);

(iv) closing for loop

(v) ; after return 0

**2)**

#include<stdio.h>

int main()

{

      int a = 0;

      printf("Enter value for a:\t");

      scanf("%d", &a);

      printf("Enter value for b:\t");

      scanf("%d", &b);

      sum\_function(a)

      return 0;

}

void sum\_function(int a, int b)

{

      int c = a + b;

      printf(Value for c: %d, c);

}

Ans:

(i) didn’t declare value b

(ii) sum\_function(…) less parameters

(iii) ending semicolon for (ii)

(iv) sum function declared after main()

(v) printf(“…..”); missing quotes

**3)**

#include <stdio.h>

int hcf(int n1,n2)

int main()

{

int n1, n2;

printf("Enter two positive integers: ");

scanf("%d %d", &n1, &n2);

printf("G.C.D of %d and %d is %d.", n1, n2, hcf(\*n1,\*n2));

}

int hcf(int \*n1, int \*n2);

{

if (n2 != 0)

return hcf(n2, n1%n2);

else

return n1;

}

Ans:

(i) int hcf(int n1,int n2)

(ii) semicolon after (i) statement

(iii) print(“……”,hcf(n1,n2));

(iv) int hcf(int n1,..)

(v) int hcf(..,int n2)

**4)**

#include<stdio.h>

int main()

{

int arr[20],i;

char demo\_c = "C";

char demo\_s[20] = 'bugHunter';

char demo\_str[20] = 'Tecstasy';

printf("Character:\t%c", &demo\_c);

printf("Character:\t%s", demo\_s);

printf("Character:\t%s", demo\_str);

while(demo){

for(i=0;i<10;i++){

arr[i]=i;

}

printf("%d",arr[i]);

}

Ans:

(i) demo\_s=” “. Double quotes

(ii) demo\_str=” “. Double quotes

(iii) while(demo). Demo undeclared

(iv)missing paren }

(v) missing return

**5)**

#include <stdio.h>

#include <stdlib.h>

#include <limits.h>

struct Stack

{

int top;

unsigned capacity;

int\* array;

};

struct Stack createStack(unsigned capacity)

{

struct Stack\* stack = (struct Stack\*) malloc(sizeof(struct Stack));

stack->capacity = capacity;

stack->top = -1;

stack->array = (int\*) malloc(stack->capacity \* sizeof(int));

return stack;

}

int isFull(struct Stack\* stack)

{ return stack->top == stack->capacity - 1; }

int isEmpty(struct Stack\* stack)

{ return stack.top == -1; }

void push(struct Stack\* stack, int item)

{

if (isFull(stack))

return;

stack->array[++stack->top] = item;

printf("%d pushed to stack\n", item);

}

int pop(struct Stack\* stack)

{

if (isEmpty(stack))

return INT\_MIN;

return stack->array[stack->top--];

}

int main()

{

struct Stack\* stack = createStack(100);

push(stack, 10);

push(stack, 20);

push(stack, 30);

printf("%d popped from stack\n", pop());

return 0;

}

Ans:

(i) struct Stack\* createStack(unsigned capacity)

(ii) int isEmpty(struct Stack\* stack)

{ return stack.top == -1; }

**IT’S stack->top**

(iii) in main while calling pop() args should be given

**6)**

#include <stdio.h>

#include <stdlib.h>

int main()

{

FILE fptr1, \*fptr2;

char filename[100], c;

printf("Enter the filename to open for reading \n");

scanf("%s", filename);

fptr1 = fopen(filename, "r+");

if (fptr1 == NULL)

{

printf("Cannot open file %s \n", filename);

exit(3);

}

printf("Enter the filename to open for writing \n");

scanf("%s", filename);

fptr2 = fopen(filename, "w");

if (fptr2 == NULL)

{

printf("Cannot open file %s \n", filename);

exit(0);

}

c = fgetsc(fptr1);

while (c != eof)

{

fputc(c, fptr2);

c = fgetc(fptr1);

}

printf("\nContents copied to %s", filename);

fclose(fptr1);

fclose(fptr2);

return 0;

}

Ans:

(i) FILE \*fptr1

(ii) c = fgetc(fptr1);

(iii) while (c != EOF)